BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 598 513 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 93308629.0

(51) Int. Cl.5: G06F 3/12, G06F 13/12

(22) Date of filing: 28.10.93

(30) Priority: 18.11.92 US 978369

(43) Date of publication of application:
25.05.94 Bulletin 94/21

(84) Designated Contracting States:
DE FR GB IT

(71) Applicant: CANON INFORMATION SYSTEMS, INC.
3188 Pullman Street
Costa Mesa, CA 92626(US)

(72) Inventor: Russell, William C.
24901 Los Gatos Drive
Laguna Hills, California 92653(US)
Inventor: Kraslavsky, Andrew J.
90 Timbre
Rancho Santa Margarita, California 92688(US)
Inventor: Wadsworth, Robert D.
1106 E. Buckingham Drive
Costa Mesa, California 92626(US)
Inventor: Barrett, Lorraine F.
21065 Castlerock Road
Yorba Linda, California 92686(US)
Inventor: Kalwitz, George A.
209 Loyola Road
Costa Mesa, California 92626(US)
Inventor: Ip, Tony K.
21041 Wheaton Terrace
Lake Forest, California 92630(US)
Inventor: Kuver, Walter D.
24766 Clarington Drive
Laguna Hills, California 92653(US)

(74) Representative: Beresford, Keith Denis Lewis et al
BERESFORD & Co.
2-5 Warwick Court
High Holborn
London WC1R 5DJ (GB)

(54) Method and apparatus for interfacing a peripheral to a local area network.

(57) Method and apparatus for interfacing a printer to a local area network includes the use of an interactive network board coupling the printer to the LAN. A bi-directional printer interface is disposed on the board and transmits print data to the printer, and receives printer status data from the printer. A LAN interface is disposed on the board for receiving print job information and printer status requests from the LAN, and for transmitting printer status information to the LAN. A ROM is disposed on the board and stores (i) application programs which receive the print job information and transmit the print data to the printer, and (ii) status and control programs which receive the printer status requests from the LAN, receive printer status data from the printer, transmit the printer status information to the LAN, and receive control information from the LAN and transmit it to the printer. A processor disposed on the board executes both the application programs and the status and control programs. Preferably, the application programs are selectively capable of placing the network board into a plurality of different printer configurations.

EP 0 598 513 A1

The present invention relates generally to a circuit board which is coupled to a local area network peripheral (e.g. a printer) and which allows the peripheral to be an intelligent, interactive network member eliminating the necessity of dedicating a personal computer to manage the peripheral. More particularly, the present invention relates to an interactive network board coupled between a responsive peripheral and a local area network.

Local Area Networks ("LANs") are known for coupling together a plurality of personal computers with peripheral devices such as printers, copiers, etc., to provide for enhanced communication and shared resources. Heretofore, peripherals such as printers coupled to a LAN were rather unintelligent, merely accepting information from the LAN and printing such information on a hard copy. Moreover, such printers usually required a host personal computer ("PC") to effectively manage the flow of data to the printer, i.e., to act as a "server" for the printer. This almost always required that the host PC be dedicated solely to the printer server task.

A number of products have recently appeared which ostensibly eliminate the need for such a dedicated PC by incorporating hardware and software into a circuit board which may be coupled into the peripheral in order to perform limited server functions. For example, ASP Computer Products, Inc. provides a device known as "JetLAN/P" which acts as a stand-alone print server for Novell networks. The JetLAN/P® device couples to a LAN using a 10Base-2 thin coaxial cable or a 10Base-T twisted-pair cable. However, the JetLAN/P® couples to the printer only through the printer's parallel port. Thus, while print information can be sent to the printer, the amount of printer status information which can be returned from the printer is severely restricted. For example, such a device may obtain "off-line" and "out of paper" status from the printer, but little else. Such a device does very little toward making the printer a truly intelligent, responsive member of the network.

Other known devices for coupling a printer to a LAN include the Hewlett-Packard Jet Direct® C2071A/B and C2059A, the Extended Systems EtherFlex®, the Intel NetPort® and NetPort II®, the Castelle LAN-Press® and JetPress®, and the MiLAN FastPort®. However, all of these devices suffer from the same disadvantages as the ASP JetLAN in that they do not allow the printer to transmit sufficient amounts of data to the LAN to enable the printer to be an effective and intelligent member of the network.

The present invention addresses the drawbacks noted above by providing structure and function on a circuit board coupled to a peripheral which will permit the peripheral to be a responsive, intelligent member of a network.

According to a first aspect of the present invention, apparatus for networking a peripheral to a local area network includes a circuit board coupled to the peripheral. A LAN interface is coupled to the board and receives peripheral job information from the LAN, receives peripheral status requests from the LAN, and transmits peripheral status information to the LAN. A peripheral interface is coupled to the board and transmits peripheral job data to the peripheral, transmits peripheral status queries to the peripheral, and receives peripheral status data from the peripheral. A PROM is coupled to the board and stores an application module which causes the peripheral job information received over the LAN to be transmitted to the peripheral as the peripheral job data. The PROM also stores a control module which causes the peripheral status requests received over the LAN to be transmitted to the peripheral as the peripheral status queries and which causes the peripheral status data received from the peripheral to be transmitted to the LAN as the peripheral status information. A RAM is coupled to the board and temporarily stores, during execution, the application module and the control module. Finally, a processor is coupled to the board for executing the application module and the control module.

According to another aspect of the present invention, a method of interfacing a peripheral to a LAN comprises the steps of coupling an interactive network board to the peripheral and to the LAN, and receiving, at the board, peripheral job information and peripheral status requests from the LAN. Then, peripheral job data is transmitted from the board to the peripheral, and peripheral status data is received at the board from the peripheral. The board also transmits peripheral status information to the LAN. A PROM disposed on the board stores a job application module and a status module. A processor on the board executes the job application module to provide the peripheral job data to the peripheral, and executes the status module to provide the peripheral status information to the LAN.

BRIEF DESCRIPTION OF THE DRAWINGS

The above-noted advantages and features of the present invention will become more readily apparent from the following detailed description of exemplary embodiments when taken in conjunction with the Drawings in which:

FIG. 1 is a block diagram of a Local Area Network according to the present invention;

In more detail, the network depicted in FIG. 1 may utilize any network software such as Novell or Unix software in order to effect communication among the various network members. The present embodiments will be described with respect to a LAN utilizing Novell NetWare® software (to be discussed in greater detail in section 3a below) although any network software may be used. A detailed description of this software package may be found in the publications "NetWare® User's Guide" and the "NetWare® Supervisor's Guide" by M&T Books, copyrighted 1990, incorporated herein by reference. See also the "NetWare® Print Server" by Novell, March 1991 edition, Novell Part No. 100-000892-001. Briefly, the file server 30 acts as a file manager, receiving, storing, queuing, caching, and transmitting files of data between LAN members. For example, data files created respectively at the PCs 10 and 12 may be routed to the file server 30 which may order those data files and then transfer the ordered data files to a printer 24 upon command from a print server in PC 18. The file server 30 may include or may be coupled to a large capacity storage member such as a 10 Gigabyte hard disk subsystem. Furthermore, the printers 32 and 34 may be coupled to the file server 30 to provide additional printing stations, if desired.

While personal computer equipment is illustrated in FIG. 1, other computer equipment may also be included, as appropriate to the network software being executed. For example, Unix workstations may be included in the network when Unix software is used, and those workstations may be used in conjunction with the illustrated PC's under appropriate circumstances.

PCs 10 and 12 may each comprise a standard work station PC capable of generating data files, transmitting them onto the LAN, receiving files from the LAN, and displaying and/or processing such files at the work station. The PCs 10 and 12, however, are not capable of exercising control over LAN peripherals (unless the network administrator is logged into that PC).

A PC capable of exerting limited control over LAN peripherals is PC 22 which includes an embedded RPRINTER program. The RPRINTER program is a MS-DOS Terminate and Stay Resident ("TSR") program which runs on a work station to allow users to share the printer 24 connected to the work station. RPRINTER is a relatively unintelligent program that does not have the ability to search printer queues for work. RPRINTER gets its work from a PSERVER (to be discussed below) that is running elsewhere in the network. Because they communicate with the attached printer over the printer's parallel port, RPRINTERs are able to obtain only limited status and to return that status information to the responsible PSERVER over the LAN 6. From a control standpoint, an RPRINTER allows stopping of a print job and little more. Some printers include RPRINTER features by offering internal or external circuit boards that provide the same limited features of the RPRINTER TSR program running in a personal computer.

Another network entity capable of exercising limited control over LAN peripherals is a printer 16 with attached circuit board 36 having an embedded QSERVER program. Here, the QSERVER program runs inside an HP LaserJet III® SI printer, and has the capability of searching the file server 30 print queues for eligible print files. The QSERVER's search queues cannot be dynamically altered nor does the QSERVER respond to any form of status inquiry. The benefit of the QSERVER is its ability to autonomously search for work. The QSERVER does not require a PSERVER running elsewhere in the system to feed it work. Since the QSERVER does not have a corresponding PSERVER and it does not itself have any status and control capabilities, it offers less control than even the RPRINTER. A QSERVER also differs from a PSERVER in that it has extremely limited notification features and cannot print banners at the beginning of each print job.

Another network member having a QSERVER capability is printer 26 which is coupled to the LAN 6 through an external NetPort device 28.

Other peripheral server programs may be executed to service various peripherals, such as scanners, copiers, facsimiles etc., and servers may also be provided based on network software protocol such as a Unix-compatible Line Printer Remote server ("LPR").

A LAN member capable of exercising significant control over LAN peripherals is the PC 18 having a PSERVER program embedded therein. PSERVER has the ability to service multiple user-defined print queues, perform dynamic search queue modification, and provide defined notification procedures for exception (failure) conditions and status and control capabilities. PSERVER is provided in several forms. PSERVER.EXE is a program that runs dedicated on a work station and controls both local and remote printers. The local printers can be connected to either serial or parallel ports, and the remote printers are printers running elsewhere in the system. Two other forms of the PSERVER program are the PSERVER.VAP and the PSERVER.NLM. These are PSERVER versions that run on the file server 30 itself. The .VAP version is for NetWare® 286, and the .NLM version is for NetWare® 386. While the PSERVER provides much more capability than the RPRINTER and QSERVER, one of its drawbacks is that the .EXE version requires a dedicated personal computer.

A dedicated personal computer running PSERVER.EXE can control as many as 16 local/remote printers and can request print information from many file server queues. However, there are several drawbacks to

While the present invention offers unique advantages on the LAN 6, these advantages are also realized when the LAN 6 is coupled to one or more other LANs in a Wide Area Network ("WAN"). FIG. 2 depicts such a WAN which includes a first LAN 41 including a server S1 40, PC's 42, 44, and 46, and a printer 48. The server S1 40 is coupled to a backbone 50 over a bus 52. The backbone 50 is nothing more than an electrical connection between a plurality of buses. Also connected to the WAN is a second LAN 61 comprising server S2 60, PC's 62, 64, and 66, and a printer 68. Server S2 60 is coupled to backbone 50 over bus 54.

The WAN may also include a remote LAN 71 comprising server S3 70, PC's 72, 74 and 76 and a printer 78. Since the LAN 71 is remote from the remainder of the system, it is coupled to backbone 50 through a bus 56, a transponder (which may include a modem) 58, and a communication line 59.

In such a WAN, assume that PC 42 is a PSERVER requesting the use of printer 78. If the printer 78 is equipped with a NEB according to the present invention, a direct communication link can be established between the PC 42 and the printer 78 whereby job information can be sent to printer 78, and status and control information can be sent from printer 78 to the LAN 41. Therefore, the NEB according to the present invention achieves its enhanced functionality even when installed in a peripheral coupled to a WAN.

FIG. 3 is a block diagram depicting the connection of the NEB 2, according to the present invention, with the printer 4 and the LAN 6. The NEB 2 is directly connected to the LAN 6 via LAN interface 101, and also to the printer 4 via a bi-directional interface, here a Small Computer System Interface ("SCSI") 100. The SCSI interface 100 is coupled to an SCSI bus 102 of the printer 4.

The NEB can also service additional SCSI devices, such as other printers (RPRINTERs) or other peripherals, daisy-chained on the SCSI bus using standard SCSI connectivity protocol. Also, the NEB may be used to drive other peripherals across the LAN itself.

The printer 4 is preferably an open-architecture printer including the SCSI bus 102 and SCSI interfaces 104 and 106. Printer 4 also includes a processor 108 such as a REDUCED INSTRUCTION SET COMPUTER ("RISC") which communicates with a RAM Memory 110 and with a printing engine 112 which actually drives the printing mechanism. The RISC processor also communicates with NVRAM 111 for storing information which needs to be maintained between power cycles, such as user-defined information, and with ROM 113 from which RISC processor 108 executes printer control. The printer 4 may also include a hard disk 114 capable of holding large amounts of data in a non-volatile way. Printer 4 also has a front panel display 116, and a keyboard 115 for inputting control commands to the printer.

Preferably, the printer 4 includes an open architecture which takes advantage of the bi-directional nature of the SCSI interface 100 to provide a great deal of status (and other) information from the printer 4 to the LAN 6 via the NEB, and also to allow fine control of the printer from a remote location. For example, such open architecture when used with the bi-directional SCSI interface permits most or all of the information on the front panel display 116 of printer 4 to be exported to a remote location, and also permits most or all of the control functions of the printer front panel keyboard 115 to be activated from the remote location.

Briefly, the open-architecture printer 4 comprises four major subsystems: Communication; Job Pipe; Page Layout and Raster functions; and Systems Services. The Communication subsystem handles the different communication devices and initiates the start of a job application. When the printer starts to receive data, the Communication subsystem sends the first part of the incoming data to each emulator for examination. The first emulator that can process the data becomes the Job Pipe driver. The system then constructs a Job Pipe to process the data (data flows into one end of the pipe, and page images flow out of the other end). This Job Pipe comprises many segments one of which is the Job Pipe driver.

The Job Pipe subsystem has a Pipe driver segment (the application for an emulator) and input and output segments. The input and output pipe segments have at least two other segments: for input, source and source filter segments; and for output, an output filter and a data sink. The input segment of the Communication subsystem delivers the input data which can be supplemented by information from a file system. The Pipe driver processes input and supplemental data. It also generates imaging commands and page layout information that it sends to the output segment. The Pipe driver may store this information to the printer disk (if present). The output segment sends this data to the Page Layout and Raster subsystem.

The Page Layout and Raster subsystem takes the imaging and page layout information and converts it to a raster image for the print engine 112. This section operates completely separately from Job Pipe.

The System Services subsystem provides file system access, console access, font services, basic system services, and image generation services. Therefore, a printer 4 having such an open architecture will take full advantage of the intelligent, interactive NEB 2 to provide increased functionality to the printer 4 and the entire network.

Table 1

Function	Implementation	Notes
Network controller (206)	National DP 83902	With DP8392 Coax Transceiver
Ethernet Interfaces:	10Base-2 (202)	Coax connector
	10Base-T (204)	RJ45 connector
	10Base-5 (203)	DB15 connector (AUI)
Embedded Processor (216)	NEC V53	16-bit/16Mhz MPU with DMA, timers, Interrupts
EPROM (Flash) (222)	256K Bytes	Network program code, board BIOS (Basic Input Output Subsystem), diagnostics
NVRAM (220)	256 Bytes	Printer Installation configuration on Network
DRAM (220)	512K Bytes	Code execution and data buffering for exp.port
SRAM (214)	8K Bytes	Buffering of incoming Ethernet packets
SCSI Controller (224)	NCR 53C90A	30-pin, internal I/F configuration with power
MAC Address and Hardware ID PROM (232)	32 Bytes	Stores MAC address and Hardward ID information
Board Size	100 mm x 125 mm	Type 2 EXP-I/F PCB, double-sided SMT
Power	5vdc, 710 ma	DC converter on board for Ethernet +12vdc/-9vdc

Preferably, the NEB 2 is installed inside the printer 4 in an expansion or options slot. The NEB 2 is thus an embedded network node with the processing and data storage features described above.

The microprocessor 216 implements a data link layer of network packet transmission and reception. Network data transfer overhead is minimized through the use of a dedicated static RAM packet buffer 214 managed directly by the network controller 206. The microprocessor 216 accesses blocks of SRAM packet data and network messages through the network controller 206, and moves them into the large DRAM memory 220.

Blocks of print image data and control information are assembled by the microprocessor 216 for transmission to the printer 4 by the SCSI controller 224 using the SCSI transfer protocol of the printer expansion port. Likewise, printer status information is transferred from printer 4 back to the NEB 2 in SCSI block format. The SCSI controller 224 operates concurrently with the network controller 206 for increased data throughput for overall NEB performance.

The microprocessor 216 is preferably an NEC V53 chip which is a fast, highly-integrated microprocessor with a 16-bit Intel-compatible processor in support of Direct Memory Access ("DMA"), interrupt, timers and DRAM refresh control. Data bus structure on the NEB 2 is implemented 16-bits wide to take advantage of the 8-Bit/16-Bit dynamic bus sizing during microprocessor I/O transfers. Control firmware and printing application software for the microprocessor 216 are stored on the NEB 2 in EPROM 222. After power-on self-test, the firmware code is selectively moved to the higher-performance DRAM 220 for actual execution. Network and printer configuration parameters are written into NVRAM 228 when the printer is first installed onto the network. Thus, NVRAM 228 allows the NEB software to recover the installation parameters after printer power has been cycled off and on.

3. SOFTWARE

Software for the LAN comprises a combination of network software, and NEB-customized software such as NEB-embedded software and software resident on the network administrator's PC.

Table 2

Function	Implementation	Notes
NEB-specific functions in NEB EPROM	CPSERVER (92KB) CRPRINTER (40KB)	Customized Print Server Customized Remote Printer
NEB-to-Network communication in NEB EPROM	CPSOCKET (30KB)	Concurrent multi-protocol operation
NEB Environment in NEB EPROM	(15KB)	Monitor, loader, POST, etc.
Extensions to NetWare® PCONSOLE for Printer Control/Configuration in Administrator's PC 14	CPCONSOL.EXE (180KB) CPINIT.EXE (120KB)	Remote Control & Stats, Auto-Reconfiguration, Print Job Logs/Statistics

3c. NEB-Embedded Software

The software developed for the NEB 2 includes software embedded in the NEB and software loaded into the network administrator's PC 14. The NEB-embedded software provides both the NetWare®-compatible node and Netware®-compatible print services directly inside the printer 4 without the overhead of a work station PC and its DOS operating system. The NEB-embedded software comprises a plurality of application modules (CPSERVER, CRPRINTER, etc.), real-time service modules, network protocol stacks, and a MONITOR program which performs application switching, process extension, device semaphores, and shares buffer-pool management. The functionality of the NEB is determined by the types of application modules and the number of protocol stacks of network layered communication software that are configured into the NEB 2. Interaction between the printer 4 and the network is coordinated by the MONITOR program which responds to real-time events while allocating NEB processing time to each application module on a multi-tasking basis.

The NEB software functions at two layers: a "run-time" or real-time layer; and a "soft-time" or applications layer. The run-time layer is comprised of components of NEB software that respond to microprocessor interrupts. This layer services devices such as a timer, queued data transfer requests from the SCSI port, or LAN data through the protocol stack routine, and the CPSOCKET (to be discussed in section 4j below) communication mechanism.

The soft-time layer is arbited and controlled by the MONITOR program (to be discussed in section 41 below) which gets control of the NEB microprocessor 216 after all real-time events have been serviced. A non-preemptive (cooperative multi-tasking) approach is used to divide the processor between the various application modules that are loaded such that no one application module can preempt other modules by capturing the microprocessor.

The NEB EPROM 222 contains up to 256 KB of application module programs and NEB initialization code. At power-up or during a programmed reset, the NEB 2 executes a POST from the EPROM 222 before selectively transferring its EPROM code to NEB DRAM 220. If the POST is successful, the NEB 2 will load its protocol stacks and application modules into DRAM, and begin execution of its application modules.

In its basic configuration, the NEB 2 contains NetWare®-compatible application modules comprising embedded versions of two configurations: the Customized Remote Printer ("CRPRINTER"); and the Customized Print Server ("CPSERVER"). Preferably, the NEB acts in only one of these configurations at a time. Further, these application modules require that a network protocol stack be loaded and functioning within the NEB.

When configured with RPRINTER functionality, the NEB operates its printer as a slave to an external print server using a CRPRINTER module. In this configuration, the NEB exports to the LAN only limited printer status information in emulation of what the standard Novell print server expects from a standard Novell RPRINTER. However, extended status information about the printer will still be available if the CPCONSOL utility (discussed above) is executed in the network administrator's PC 14.

administrator to view network and printer status, job statistics, and a log of the previously-processed jobs and error conditions. CPCONSOL gathers the requested information by communicating with the NEB-embedded software program module CPSOCKET.

Another customized software program resident on the network administrator's PC 14 is Customized Peripheral Initializer ("CPINIT"; to be discussed in section 4h below) which is also a menu-driven DOS executable program. The function of the program is to configure, reconfigure, and initialize the printer 4 attached to the NEB 2.

The CPINIT module will configure the NEB to act as a print server with one attached printer and specifies its primary file server by which the NEB will determine which queues to service. CPINIT is the program that supervises all like-customized devices on the LAN (e.g. other NEBs installed in other open-architecture printers). CPINIT accomplishes this task by communicating over the network with other NEBs that reside within open-architecture peripheral devices. CPINIT is used to configure each NEB with the appropriate basic configuration information such as configuring the NEB as CPSERVER or CRPRINTER. The basic configuration information comprises NEB environment settings (including which print server applications are active), as well as device environment options (e.g. a list of fonts and emulations to download printer initialization time), and device default settings (such as the internal device time/date/time zone, buffer size, disk and logging information, and printer name). The CPINIT program also displays status information about the NEB (such as the firmware level loaded in the NEB and reports latent POST errors).

The CPINIT program will broadcast over the network to see which other customized devices are available on the LAN. The NEBs attached to such other customized devices will respond with their identification numbers, their device types, and their configuration states. CPINIT will construct a list of these NEBs and devices that will be presented to the netwrok administrator to allow their configuration or reconfiguration.

A DOWNLOADER program may also be loaded into the network administrator's PC 14 to download executable files to the NEB across the network (to be discussed in greater detail in section 4n below).

Another customized program which may run on the network administrator's PC 14 is CPFLASH which may be used to remotely flash new firmware into EPROM 222, as will be discussed in greater detail in section 4q below.

4. OPERATION

At first, an overview discussion of the NEB structure and functions will be provided with respect to the flowchart of FIGS. 5A, 5B and 5C. Thereafter, more detailed descriptions of various aspects of the NEB hardware and software will be provided with respect to sections 4a to 4q.

The present invention takes advantage of the bi-directional nature of the communication between the printer and the NEB, and the NEB's ability to process information on a multi-tasking basis. That is, the bi-directional SCSI bus can transmit large quantities of data both to and from the printer, enabling the NEB to receive large quantities of specific status data from the printer or even data input from the peripheral (such as image data input from a scanner). The NEB microprocessor processes information on a multi-tasking basis (sequential but shared) effectively parallel processing information received from the network and information received from the printer. This multi-tasking processing insures that the NEB is responsive to both the network and the printer on a near real-time basis.

FIGS. 5A, 5B, and 5C comprise a top-level flowchart depicting a notional sequence of events which may occur when the NEB and its associated software is installed in a printer coupled to a local area network. Overall, the printer renders print information and is coupled to the NEB through a bi-directional SCSI interface. The printer may also have a parallel port and/or a serial port for receiving print information from other sources. The NEB is connected to the printer via the bi-directional SCSI interface, the board receiving printer information from the local area network. The board sends print jobs and printer status inquiries to the printer over the SCSI interface, receives printer status from the printer over the SCSI interface, and reports printer status over the network.

Where the NEB is coupled to a data generating device such as a scanner, the board is connected to the scanner through the bi-directional SCSI interface and is coupled to the network via the LAN interface. The board receives status request information from the network and will pass this information to the scanner over the bi-directional interface. The board will also receive the data generated by the scanner over the bi-directional interface, and will pass that data onto the network over the LAN interface.

Illustrating a sequence of events which may occur when the NEB is installed in a printer, FIG. 5A begins when power is applied to the NEB at Step S1. At Step S2, the NEB executes a power-on-self-test ("POST") from EPROM 220. At Step S3, if the POST is successfully completed, the process moves to Step

allows more than one server to advertise network services at the same time on the same node. Thus, CPSOCKET and CPSERVER both advertise their services through SAPSERVER and respond to inquiries from other network applications. Since each EtherNet board can have only one SAP socket number, SAPSERVER will function to advertise both NEB identities without confusion to the LAN.

In summary, Step S13 is a method of identifying a single interactive network board as two network servers (e.g. CPSERVER and CPSOCKET) comprising the steps of transmitting to the network at a predetermined time interval a signal indicating that the board is a first type of network entity, the signal including an identification signal unique to the board; then, transmitting a second signal to the network at the predetermined time interval to indicate that the board is a second type of network entity, the second signal also including the same unique identification signal. Once a signal is received from the network requesting that the board perform functions of one of the types of network entities, direct communication is established between the board (acting as the requested type of network entity) and the network entity which generated the request. When the direct communication is established, the NEB will utilize a new unique identification signal.

At Step S14, both the LAN and the SCSI interfaces are checked for data that is being directed to CPSOCKET (to be discussed in greater detail in section 4j below). The SCSI interface will typically have printer status data which is to be passed to the LAN in response to a previously-received request for status. CPSOCKET is the NEB-resident TSR program that responds to such requests for connection, requests for data download, or requests for services from remote utilities. CPSOCKET gathers information from the NEB or the printer, as needed, monitors requests to write to the log file, monitors application requests for device status, and maintains job statistics, as discussed above.

Briefly, the CPSOCKET program is a method of interfacing an interactive network board between the network and a peripheral device, comprising the steps of transferring a program from board ROM to board RAM for execution from the RAM; and monitoring, with the program, a board network interface to detect a network communication directed to the peripheral device. The program then commands the peripheral device to perform a function in response to the network communication, and monitors a board bi-directional peripheral interface to detect and store status information of the peripheral device. Finally, the program outputs the peripheral device status information onto the network through the network interface in response to another network communication.

In FIG. 5B, Steps S15 and S17 indicate "run-time" layer functions, and Step S20 represents a "soft-time" application layer. First, Step S15 determines whether data is being received over the LAN. When LAN data is received, the process proceeds to Step S16 and the software protocol type is determined (to be discussed in greater detail in section 4f below). For example, the Ethernet data received over the LAN may be one of the following software protocols: e.g. Netware® over SPX/IPX; UNIX over TCP/IP; or Mac Systems 7 over AppleTalk. Basically, the software protocol type may be determined according to the frame packet type sensed in Step S7 above.

If CPSOCKET determines that LAN data is not being received in Step S15, Step S17 determines whether SCSI data is being received, and if SCSI data is being received, it is input from the printer in Step S18, and then stored in DRAM 220 in Step S19.

After the storing of printer data in Step S19, or if SCSI data is not being received in Step S17, the process proceeds to Step S20 where "soft-time" tasks are performed on a multi-tasking basis as controlled by a multi-tasking software program called "MONITOR" (to be discussed in greater detail in section 4l below). Step S20 is therefore a "background" process which runs concurrently throughout the flowchart depicted in FIGS. 5A, 5B and 5C. That is, whenever "soft-time" tasks are being performed, the microprocessor 216 will ensure time-shared, parallel, non-preemptive processing of the "soft-time" tasks.

More particularly, MONITOR is a software module downloaded from EPROM 222 to DRAM 220 in Step S6. MONITOR is a non-preemptive multitasking monitor which distributes the processor usage among the several application tasks which are currently active. The non-preemptive nature of the monitor requires that each application task periodically relinquish control so that other tasks gain the opportunity to execute. The relinquish control mechanism is implemented using a software interrupt to pass control to the MONITOR. At an interrupt, MONITOR saves the state of the current task, restores the state of another active task, and resumes (or commences) execution of the new task. The task which originally relinquished control eventually regains control at the interrupt point, i.e. with its context restored to the same condition as when it relinquished control.

In summary, Step S20 comprises the step of monitoring a plurality of application tasks in a multi-tasking interactive network board to distribute processor resources. A memory stores a first application task which may queue a file server to get a network interface to obtain a queue of print files to be printed, and which channels the print files to a printer coupled to the board through an interface. The memory also stores a

comprises the step of issuing, at a remote location, a command to the board that will cause the board to transfer printer status information through the board to the remote location through the LAN interface. At the remote location, a printer status may be displayed; and a second command may be issued at the remote location to the board through the LAN interface to cause the board to p rform a manually-operable function.

If the received LAN data is neither a print job nor a status request, it is determined at Step S29 that the received data may be a download operation, i.e., a transfer of data into the NEB for updating the ROM or RAM applications, e.g. download may be utilized for transient diagnostics to be run on the NEB.

First, at Step S30, the data is downloaded from the LAN to the DRAM 220 (to be discussed in greater detail in section 4n below). That is, the download is a process by which data may be loaded into a network node and then acted upon or executed. For example, anything from patch code, to manufacturing test routines, to firmware updates for the EPROM may be downloaded. Also, application modules may be stored in the LAN file server and then downloaded to the NEB every morning.

In summary, the downloading of data from LAN to DRAM comprises a method for altering an operational mode of an interactive network board having a LAN interface, including the step of activating a LAN communication program for execution from DRAM, the communication program channelling print information on the LAN to a peripheral printer. Executable instructions which correspond to the altered operational mode are then downloaded into DRAM via the LAN interface. The board is then commanded via the LAN interface to begin execution of the altered operational mode.

At Step S31, it is determined whether the downloaded information is destined for EPROM 222 or DRAM 220. If the information is destined for EPROM, a ROM image is assembled at Step S32 (to be discussed in greater detail in section 4o below). For example, downloading of EPROM firmware from a remote location provides unique flexibility. In particular, downloading of on-board test routines, and changing EPROM configuration firmware can be performed from a remote location after the board is installed in the printer.

Step S32 is the process which constructs the binary image file which is to be programmed into the EPROM 222. The data destined for the EPROM is first downloaded to DRAM 220 where a utility reads a configuration file containing the names of the modules to be placed in the ROM image. Then, a complete binary image file is constructed containing all of the specified modules. A header precedes each module in the image, the header identifying the module, describing its attributes, and pointing to the succeeding header to aid in locating the modules during loading. The last module loaded in the EPROM is the EPROM-resident code. It is placed at the end of the ROM image so that the power-up initialization code resides at the address expected by the microprocessor 216.

In summary, Step S32 comprises a process for formatting a binary image file which contains executable code modules for storage in the EPROM. First, a configuration file is read which specifies the code modules which form the binary image. Next, a header is formed for each module specified in the configuration file, the header including an identification of the module, a definition of the module's attributes, and a pointer to a header for a succeeding module. The binary image file is then constructed containing the specified modules and their associated headers. Finally, a module of ROM-resident code is appended to the binary image, the ROM-resident code receiving control at power-up, providing POST, loading at least some of the modules from the binary image file into DRAM 220, and providing basic board I/O services.

Before writing new data into EPROM 222, it is first necessary to unequivocally ensure that a write operation is, in fact, intended. Obviously, any accidental writings into EPROM 222 could render the NEB unusable. Therefore, before information may be "flashed" to EPROM 222, a specified sequence of events will occur in Step S33 in order to access the EPROM (to be discussed in greater detail in section 4p below). In the present embodiment, unless two data bits are changed in two separate I/O locations, the +12 Volts necessary to write to the EPROM will not be provided.

Briefly, a method of ensuring that the EPROM is not accidentally written into comprises a method of performing a flash operation on an EPROM resident on an interactive network board having a processing unit and a memory, including the step of sending an I/O write signal to the processing unit. Then, the processing unit generates a first address in the memory to cause a first bit to be in a predetermined state in response to the I/O signal. A power unit is then caused to provide +12 Volts to a transistor in response to the first bit being placed in the predetermined state. Then, an I/O receive signal is sent to the processing unit which generates a second address in the memory to cause a second bit to be in a preselected state in response to the I/O receive signal. Then, the transistor is turned on in response to the second bit being placed in the preselected state causing the +12 Volts to flow to a power terminal of the EPROM, allowing a write operation to take place.

Before the new ROM image is actually stored in EPROM 222, at Step S34 the new ROM image must be checksummed and verified with a checksum value sent after the ROM image is received. Prior to erasing EPROM 222, data and modules to be preserved, such as the MAC address, must be loaded into

microprocessor 216 may retrieve selective modules from EPROM 222 for loading into DRAM 220 and for execution from the DRAM.

FIG. 6 shows the sequence by which different modules are retrieved from EPROM 222 and loaded into DRAM 220. In Step S6001, microprocessor 216 loads the SCSI driver from EPROM 222 into DRAM 220. The SCSI driver provides for operational sequence and control over SCSI controller 224 and permits interface with printer 4 so as to send printer 4 print data and so as to send and receive control information to and from printer 4.

In Step S6002, microprocessor 216 loads the link support layer, or "LSL", from EPROM 222 into DRAM 220, and in Step S6003 microprocessor 216 loads network driver software from EPROM 222 into DRAM 220, and thereupon microprocessor 216 begins to execute the link support layer and the network driver from DRAM 220. The link support layer and the network driver provide common access to LAN communications on LAN bus 6. More particularly, as shown in FIG. 7, all networked devices, including a device such as NEB 2, interface with LAN bus 6 via an electrical interface 301 such as the network controller 206 used on NEB 2. The electrical interface 301 is driven by network driver 302 which in turn receives LAN frame data from link support layer software 304. Both the link support layer 304 and the network driver 302 are common to different kinds of network software. For example, as further shown in FIG. 7, network application programs, such as those provided in NetWare® software by Novell (as illustrated at Arrow A) interface with the link support layer and the network driver via an internetwork packet exchange program, or "IPX", 305 and a sequenced packet exchange program, or "SPX", 306. On the other hand, network application programs from UNIX provided by AT&T (as illustrated at Arrow B) interface to the LSL through "IP" module 315 and "TCP" module 316.

In NEB 2, only one type of network application programs is normally executed at any one time (although multiprotocol operations are possible as discussed in section 4f below). Explanation here will be made for NetWare® network application, programs although it is also possible for UNIX network application programs to be executed as well.

In Step S6004, microprocessor 216 loads a PRESCAN program from EPROM 222 and stores it into DRAM 220, and thereupon begins executing the PRESCAN program from DRAM 220. PRESCAN software interfaces with the link support layer to determine the frame packet type being transmitted on LAN bus 6. More particularly, as described above, there are four different possible frame packet types on an Ethernet-type network LAN bus: Ethernet 802.3, Ethernet II, Ethernet 802.2, and Ethernet SNAP. As described more fully below in section 4e, the PRESCAN software module monitors network communications on LAN bus 6 to determine the frame packet type. The frame packet type, once determined by PRESCAN, is stored in a predetermined common location in DRAM 220 for use by other network communication modules in the NEB. After determining the frame packet type, PRESCAN signals microprocessor 216 that its tasks are completed and allows microprocessor 216 to overwrite the memory occupied by the PRESCAN program with another program module.

In Step S6005 microprocessor 216 retrieves the IPX and SPX program modules from EPROM 222 and stores them in DRAM 220, and thereupon begins executing the IPX and SPX modules from DRAM 220. Both IPX and SPX use the frame packet type determined by the PRESCAN module.

In Step S6006 microprocessor 216 retrieves the CNETX program module from EPROM 222 and loads that module into DRAM 220 and thereupon begins execution from DRAM 220. CNETX provides localized DOS-like functionality to the NEB.

In Step S6007, microprocessor 216 loads the SAPSERVER program module from EPROM 222 into DRAM 220 and begins executing the SAPSERVER module from DRAM 220. As described more fully below in section 4g, SAPSERVER is a program module which allows two network server entities, such as CPSOCKET and CPSERVER, to advertise simultaneously from the single network node assigned to the NEB board, even though conventional network application programs such as those provided by Netware® only permit advertising of a single network server entity from each network node.

In Step S6008 microprocessor 216 retrieves the non-preemptive multi-tasking MONITOR (see section 41 below) from EPROM 222 and stores it into DRAM 220 and begins executing the multi-tasking monitor from DRAM 220.

In Step S6009 microprocessor 216 retrieves the CPSOCKET server software module from EPROM 222 and loads it into DRAM 220 and begins executing the CPSOCKET server from DRAM 220. As will be described more fully below in section 4j, CPSOCKET initiates a request to SAPSERVER to advertise on behalf of CPSOCKET, and SAPSERVER begins making SAP advertisements on LAN bus 6.

In Step S6010 microprocessor 216 retrieves print application servers such as CPSERVER or CRPRINTER from EPROM 222 and loads the print application servers into DRAM 222. In the case of CPSERVER, microprocessor 216 begins executing the loaded print application servers from DRAM 220 which in turn

itself will always activate those software modules needed to perform basic communication with the LAN. Using CPINIT, the network manager can determine, remotely, the current configuration of the NEB, or he/she can change the configuration as desired. Since the configuration information is stored in EPROM on the NEB board, the configuration information is retained across power cycles.

The process by which the software programs for a particular configuration are downloaded from the EPROM 222 to the DRAM 220 will be described below with respect to FIG. 8.

After the board has been powered up at Step S1, the process proceeds to Step S8001 where microprocessor 216 accesses EPROM-resident code in the EPROM 222 to read a configuration code (typically a word) from NVRAM 228. The configuration code will specify modules which can provide the NEB with either a PSERVER or RPRINTER functionality. Although the present embodiment includes only RPRINTER or PSERVER functional configurations, other configurations may be utilized where, for instance, the NEB 2 is installed in a different LAN entity, such as a scanner or a facsimile machine.

After reading the configuration code from NVRAM 228, the microprocessor, at Step S8002, forms a configuration mask whose bit pattern corresponds to the read configuration code. At Step S8003, a loader module resident in EPROM 222 compares the configuration mask to the plurality of firmware modules stored in the EPROM 222.

In detail, in Step S8004, a process begins whereby the EPROM-resident software modules are selected in bit-wise correspondence to the binary digits of the configuration code read from NVRAM 228. If it is determined in Step S8004 that the currently-examined bit of the bit pattern matches a stored module, then that module is selected at Step S8005) for downloading to DRAM 220, and the process skips to the next bit at step S8006. Likewise, if Step S8004 determines that a bit of the bit pattern does not match the stored module, the process skips to the next bit at Step S8006.

At Step S8007, it is determined whether the bit tested in Step S8004 is the last bit of the configuration mask bit pattern. If the tested bit is not the last bit, the process loops back to Step S8004, where the next bit of the bit pattern is tested with respect to the next stored module. When the last bit of the configuration mask bit pattern has been tested, the selected software modules are downloaded from EPROM 222 to DRAM 220 at Step S8008.

In the present embodiment, the software modules are loaded in the following sequence: SCSI Driver; Link Support Layer; Network Driver; Prescan; IPX/SPX; CNETX; SAPSERVER; MONITOR; CPSOCKET; and Print Applications (e.g CPSERVER, CRPRINTER) (see FIG. 6).

After all of the software modules which correspond to the configuration codes stored in NVRAM 228 have been downloaded to DRAM 220, the loader function will pass program execution control to the MONITOR multi-tasking program at Step S8009.

As discussed earlier, the configuration code stored in NVRAM 228 may be remotely altered using CPINIT. This provides greater flexibility for making minor modifications to CPSERVER or CRPRINTER, or where entirely new configurations are desired to be set. Therefore, at Step S8010, a new configuration is received over LAN 6, and is loaded into NVRAM 228 at Step S8011. Preferably, the old configuration code will be erased or overwritten with the new configuration code. Then, the NEB reboots itself and returns to Step S1.

4e. Determining Frame Packet Type Using PRESCAN

On any local area network, data is transmitted between network devices in packets or frames. But even in the context of a common network architecture, such as Ethernet, more than one format for the frame may be supported. Thus, even though it is known that Ethernet architecture is being used, it is not possible to determine the arrangement of data within each physical frame or packet of information on the Ethernet bus. In particular, as described above, Ethernet supports four arrangements of data, or formats, as follows: Ethernet 802.3, Ethernet II, Ethernet 802.2, and Ethernet SNAP.

In conventional network devices, which provide for a manually selectable operator interface, it is possible to advise the network device of the particular frame type being used on the Ethernet network. In the context of NEB 2, for which operator access is provided only via the network interface (or via the serial port 218 in a test configuration), it is not possible to set the frame packet type without first allowing operator access to the local area network which, of course, requires knowledge of the frame packet type.

The PRESCAN software module allows NEB 2 to automatically determine the frame packet type currently being used for LAN communication on the LAN bus by monitoring broadcast communications on the LAN bus until the proper frame packet type is recognized. PRESCAN makes this determination based on recognizable components that are common to all four frame packet types used on Ethernet.

as the aforementioned CPSERVER, which checks job queues in a file server on a Novell operating system, as well as a UNIX-compatible peripheral server, such as the aforementioned CLPR (Custom Line Printer Remote), which, in coordination with checks made by CPSERVER, also checks for job queues in a file server for a UNIX operating system. Both servers, here CPSERVER and CLPR, service common peripheral resources, here a single peripheral such as a printer, and to avoid contention for control of the common resources, both servers are able to seize control of the peripheral to the exclusion of other servers, to signal other servers that control has been seized, and to relinquish control of the peripheral when the job queue has been emptied. It is also possible for each server to check with other servers to determine if other servers have a pending request for use of the peripheral. In the case where there is a pending request, the server can relinquish control of the peripheral at the end of a current job even though there are jobs remaining in the job queue, so as to allow alternating use of the peripheral by each server.

FIG. 11 illustrates NEB 2 configured for multiprotocol network operations. FIG. 11 illustrates a combined Novell/UNIX multiprotocol environment, but it is to be understood that other operating protocols may be substituted for or used in combination with those shown in FIG. 11. In FIG. 11, NEB 2 is interfaced to LAN bus 6 via electrical interface 321, network driver 322, and link support layer ("LSL") 324, much as illustrated above in FIG. 7. Novell-specific operating protocols are indicated at reference numerals 325, 326 and 327. More specifically, 325 and 326 are an SPX/IPX operating protocol stack (or tower) by which Novell-compatible applications programs communicate with the LAN bus through LSL. Novell-compatible applications programs, including a Novell-compatible server such as CPSERVER, are illustrated at 327. The Novell-compatible software drives printer 4 via bi-directional SCSI bus 102 as described above.

UNIX-compatible operating protocols are illustrated at reference numerals 335, 336 and 337. More specifically, 335 and 336 comprise a TCP/IP operating protocol stack (or tower) by which UNIX-compatible application programs communicate to LAN bus 6 via LSL. UNIX-compatible network application programs, including a UNIX-compatible printer server such as CLPR, are designated at 337. The print server CLPR drives printer 4 via SCSI bus 102 as described above.

Prescan module 339 interfaces with LSL 324 to determine the frame packet type being transmitted on LAN bus 6 for each of the operating systems. In more detail, each operating system such as the UNIX operating system and the Novell operating system can communicate on LAN bus 6 in a variety of frame packet types. When LAN bus 6 is an Ethernet type LAN bus, then a UNIX operating system can communicate over the Ethernet by any of three frame packet types, namely, Ethernet 802.2, Ethernet II and Ethernet SNAP. Likewise, when LAN bus 6 is an Ethernet-type bus, then a Novell operating system can communicate over the LAN bus by any of four frame packet types, namely, Ethernet 802.2, Ethernet 802.3, Ethernet II and Ethernet SNAP. It is possible for both the Novell operating system and the UNIX operating system to use the same frame packet type; it is the operating system protocols (SPX/IPX for Novell and TCP/IP for UNIX) which determine which one of the operating systems in a multiprotocol environment is currently communicating on the LAN bus.

In the multiprotocol environment illustrated in FIG. 11, PRESCAN module 339 determines the frame packet type being used by each operating system by repeating the steps shown in FIG. 10 for each of the operating system protocols (see section 4e above). For example, when Novell-compatible and UNIX-compatible systems comprise the multiprotocol environment, then PRESCAN simultaneously binds through LSL to all four frame packet types for an SPX/IPX protocol tower, so as to determine the frame packet type in accordance with the data group returned from LSL which has the proper IPX header. Then, PRESCAN binds simultaneously through LSL through all three frame packet types having a TCP/IP protocol tower. PRESCAN determines the frame packet type being used by the UNIX-compatible operating system in accordance with the data group having the proper TCP/IP header.

In more detail, to adaptively and automatically determine which of plural predetermined frame packet types is currently being used for LAN communication in a multiprotocol network environment, the PRESCAN program module 339 is downloaded from EPROM 222 into DRAM 220 where microprocessor 216 executes the PRESCAN module. To determine the frame packet type for the first operating system, PRESCAN first configures LSL to bind simultaneously to a plurality of frame packet types corresponding to a first operating system protocol, such as SPX/IPX operating protocol for Novell-compatible operating systems. Network driver 322 monitors the LAN communication bus to capture broadcast traffic for the first operating system. In response to capturing such broadcast traffic, LSL provides plural data groups for the captured broadcast traffic, each of the data groups corresponding to a different one of the plural packet types. The PRESCAN module 339 is reactivated to prescan each data group for the presence of a predetermined header, such as the SPX/IPX header; and stores the frame packet type corresponding to the data group having the predetermined header for use by the first operating protocol tower.

As shown in FIG. 13, after microprocessor 216 retrieves the SAPSERVER program module from EPROM 222 and stores it in DRAM 220, microprocessor 216 commences operation of the SAPSERVER program and configures it to listen for SAP proprietary broadcasts to the SAP proprietary socket (Step S1301). In Step S1302, after microprocessor 216 retrieves the CPSOCKET module from EPROM 222 and stores it for execution in DRAM 220, the CPSOCKET program module issues a request to SAPSERVER to advertise CPSOCKET services. In accordance with standard SAP protocol, SAPSERVER commences periodic advertisements for CPSOCKET (Step S1303), for example, at one minute intervals.

In Step S1304, after microprocessor 216 retrieves the CPSERVER module from EPROM 222 and stores it for execution from DRAM 220, CPSERVER issues a request to SAPSERVER for SAPSERVER to advertise CPSERVER services over the network. SAPSERVER commences periodic SAP advertisements for the services of CPSERVER and also continues to advertise the services for CPSOCKET. As shown in Step S1305, the advertisements are interleaved.

Step S1306 determines whether a broadcast request has been received at the SAP proprietary socket (e.g. socket number 453). Until a broadcast request has been received at the proprietary socket, SAPSERVER simply continues to interleavedly advertise for the services of CPSERVER and CPSOCKET. However, when a broadcast request is received at the proprietary socket then in Step S1307 SAPSERVER determines whether the broadcast request is for the services of one of its clients, here for the services of CPSOCKET or CPSERVER. If the broadcast request is not for one of SAPSERVER's clients, then flow simply returns to Step S1305 where SAPSERVER continues to interleavedly advertise for its clients. On the other hand, if the broadcast request is for one of SAPSERVER's clients, then flow advances to Step S1308.

In Step S1308, SAPSERVER responds with an IPX packet on the proprietary socket number 453. The IPX packet contains the server type of its client, the server name, and a communication socket number. The IPX packet also designates a communication socket over which the broadcast requestor can establish direct communication with the client. Thereupon, SAPSERVER returns to Step S1305 so as to continue to interleavedly advertise for each of its clients.

In Step S1309, the broadcast requester establishes direct SPX connection with the client designated in the broadcast request over the communication socket designated in Step S1308. In the present configuration, where services of the print server CPSERVER is requested, the socket number is 8060. On the other hand, when requests for services of the CPSOCKET server is requested, the socket number is 83B4 for communication and 83B5 for connection. Direct communication then proceeds as described more fully hereinbelow.

4h. Configuring The Networked Printer Using CPINIT

FIG. 14 is a flow diagram showing how a network administrator can use CPINIT from PC 14 to initialize and to configure, and later to reconfigure, both NEB 2 and printer 4 in which the NEB resides.

In Step S1401, the CPINIT utility uses a service advertising protocol (SAP) on the network to determine which networked printer devices are available to respond to CPINIT inquiries. At each of the NEB boards, CPSOCKET responds with server type, server name and a unique socket number by which each NEB can be accessed directly, and an indication of whether or not the NEB requires configuration.

In Step S1402, CPINIT constructs a list of all NEB's and their associated devices, and presents them in a menu form so that they can be selected by the system administrator. Following selection, CPINIT requests the current configuration of the targeted NEB (Step S1403). More specifically, CPINIT sends a request to the targeted NEB via the LAN interface. At the NEB, CPSOCKET, receives the request for configuration information from the LAN interface. CPSOCKET collects the needed configuration information, and directs it via the LAN interface to CPINIT at the system administrator's PC 14 (Step S1404). In Step S1405, CPINIT displays a menu of the current configuration of the targeted NEB.

In Steps S1406 through S1408, the system administrator specifies a desired configuration for the targeted board. More particularly, configuration is specified on the system administrator's PC 14 by means of a user interface such as a menu display. The following configuration parameters are selected by the operator to set the configuration information: (1) logging information (Step S1406), (2) NEB name (Step S1407), and (3) application type (such as CPSERVER) (Step S1408).

Under logging information, the system administrator specifies one of four different levels of logging: "NONE", in which logging is disabled; "AUTO", in which basic printer usage statistics are logged once per day; "ERROR", in which basic printer usage statistics and error events are logged as they occur; and "JOB", in which basic usage printer statistics, error events and job start/end information are all logged as they occur. After selecting the log preference, the system administrator must also set the maximum log size (except when "NONE" is selected) so as to permit the printer to reserve this amount of space on its disk (or

into four groups: Common Environment, Interface, Control, and Quality.

Upon selecting Common Environment, CPCONSOL will initiate a LAN request to the target NEB for the settings for Emulation Mode, Feeder, and Total Page Count. CPSOCKET at the target NEB will receive the LAN request, obtain the desired information from its attached printer via the bi-directional SCSI interface, and send the information to CPCONSOL at the administrator's PC 14 via the LAN interface. There, CPCONSOL displays a listing showing the emulation mode, the feeder, and the total page count.

Upon selecting the Interface menu CPCONSOL will initiate a LAN request to the targeted NEB for interface information. When the NEB responds, CPCONSOL causes the interface listing to display the interface that is currently set for the selected printer.

Selecting the Control menu likewise causes CPCONSOL to initiate a LAN request to the targeted NEB which in turn interrogates its printer via the bi-directional SCSI bus for printer settings. The printer settings are returned to CPCONSOL over the LAN interface which displays the current settings of the printer in accordance with Table 3.

Table 3

Control Information	Description
Contrast	Printer contrast setting.
Timeout	This is the setting of the job time-out set in the printer.
Message	Language in which messages are displayed.
Copy	Number of copies of each page to be printed.
Offset X	The offset, if any, in the horizontal direction from the upper left corner of the page, in millimeters.
Offset Y	The offset, if any, in the vertical direction from the upper left corner of the page in millimeters.
Error Skip	Displays whether the printer is set for automatic or manual error skipping.
Buzzer	On or Off setting of the printer buzzer.
Toner Low	If the toner low, a WARNING is displayed.
28-Error	Detection of Memory Full error can be turned on or off.
Paper	Paper sizes that are available in the printer.
Current Paper	Paper cassette that is currently selected in the printer.

Upon selecting the Quality group, CPCONSOL, after requesting and receiving information from the targeted NEB via the LAN interface, displays the settings for Selection Mode, Refine, Memory Usage and Low Resolution mode.

[Network Group (Step S1508)]

The network selection allows CPCONSOL to display the compiled statistics about the networked printer performance on the network (Step S1509), and to modify and store the new network group (Step S1510). These are subdivided into media-dependent and media-independent related transmit and receive statistics. CPCONSOL can also clear all statistics.

When the system administrator selects the Network group, CPCONSOL initiates a network request via the LAN interface to the targeted NEB. At the NEB, CPSOCKET, responds to the request and obtains the needed performance information. The information is collected by CPSOCKET and returned to CPCONSOL at the administrator's PC 14 via the LAN interface. At the administrator's PC 14, CPCONSOL displays the media-dependent and media-independent transmit and receive information.

[Logging Group (Step S1511)]

The logging group selection allows CPCONSOL to display the set of job-related statistics that the NEB compiles (Step S1512), and to modify and store the new logging group (Step S1573). The displayed data include job averages, page averages, and performance data. CPCONSOL can reset the totals to zero with this menu as well. In addition to statistics, the NEB can create a log for every print job, write the log to a work station disk, or clear the log file, as configured by CPINIT.

If the system administrator selects the Logging Group option, then CPCONSOL directs a LAN request for the log file to the targeted NEB via the LAN interface. At the NEB, CPSOCKET receives the request and, since CPSOCKET stores the log file on the printer, requests the log file from the printer via the bi-directional SCSI interface. The NEB retrieves the log file from wherever it is stored (such as its disk 114) and provides the file to CPSOCKET via the bi-directional SCSI interface. CPSOCKET then puts the log file onto the network via the LAN interface for receipt by CPCONSOL.

The log file includes values for the statistics which are divided into three categories: Daily, Cumulative, and Average. Daily shows the values for the current day. Cumulative shows the totals for all days since last reset, or since power-on for a printer without a disk drive. Average is the cumulative totals divided by the number of days since the last reset. For each of the three categories, the NEB maintains totals for the following values (unless CPINIT has set the logging level to "NONE"): days (number of days since a reset was issued or since power-on), pages printed, print jobs processed, off-line time, and printing time.

CPCONSOL also retrieves the stored log file to the screen for viewing and printing. The log file is in reverse chronological order and includes the following record types. The precise content of the log file varies in accordance with the logging level set by CPINIT, as summarized in Table 7.

Table 7

Type	Data	Description
STD	⟨Days⟩⟨Pages⟩⟨Jobs⟩⟨Offline⟩⟨Printing⟩	daily statistics
STC	⟨Days⟩⟨Pages⟩⟨Jobs⟩⟨offline⟩⟨Printing⟩	cumulative statistics
STA	⟨Days⟩⟨Pages⟩⟨Jobs⟩⟩Offline⟩⟨Printing⟩	average statistics
SOJ	⟨Application⟩⟨User⟩⟨Job⟩⟨File server⟩ ⟨Queue⟩⟨Form⟩	start of job
INI	⟨NEB Type⟩⟨ROM/MAC Address⟩⟨Printer Name⟩	Initialization record
POW	⟨NEB Type⟩⟨ROM/MAC Address⟩⟨Printer Name⟩	power on record
RBT	⟨NEB Type⟩⟨ROM/MAC Address⟩⟨Printer Name⟩	reboot record
WAR	⟨Application⟩⟨Warning⟩	warning
EOJ	⟨Application⟩⟨User⟩⟨Job⟩⟨Disposition⟩	end of job
ERR	⟨Application⟩⟨Error⟩	error

[Application Control (Step S1514)]

Application control allows CPCONSOL to view the current configuration of the NEB within the network (as either CPSERVER or CRPRINTER) (Step S1515) and to activate/deactivate or modify and store that application (Step S1516). Access to the targeted NEB is provided via the LAN interface which responds to the CPCONSOL request by putting a result code on the LAN interface.

[Printer Status (Step S1517)]

This menu allows CPCONSOL to display the current status of the printer attached to the NEB (Step S1518), and to modify and store the new printer status (Step S1519). CPCONSOL directs a status request to the targeted NEB via the LAN interface. At the targeted NEB, CPSOCKET receives the status request and sends a request for the needed status information to the printer via the bi-directional SCSI interface. CPSOCKET receives the status information from the printer over the bi-directional SCSI interface and

CPCONSOL and DOWNLOADER.

CPSOCKET is responsible for the internal configuration of the NEB, such as configuration as either a PSERVER or an RPRINTER. Configurations are set at the request of CPINIT, as described above, but it is CPSOCKET that receives those configuration commands and physically alters NVRAM 228.

CPSOCKET also maintains a table of default settings for the device environment (that is, a guaranteed safe environment, see section 4m below), downloads the basic configuration information for the printer and for the NEB (for example, fonts and emulations) at device power-up (see section 4d above), provides device status information, statistics, and log information in response to CPCONSOL requests, and provides reset, re-boot, and firmware download capabilities.

FIGS. 16A and 16B comprise a detailed flow diagram showing operation of the CPSOCKET program. In Step S1601, after successful power-on-self-test (POST), microprocessor 216 transfers the CPSOCKET program module from its storage locations in EPROM 222 into appropriate storage locations in DRAM 220. During transfer, microprocessor 216 configures the CPSOCKET program in accordance with the configuration information for the CPSOCKET program stored in NVRAM 228. Thus, for example, it is possible to selectively activate certain portions of the CPSOCKET program module in accordance with desired levels of complexity, those desired levels of complexity being stored in NVRAM 228.

In Step S1602, the NEB commences execution of the CPSOCKET from DRAM 220. CPSOCKET is executed in a multi-tasking soft-time environment by the non-preemptive MONITOR which permits non-preemptive execution of other application programs such as CPSERVER without letting one application program seize control of the microprocessor to the exclusion of other application programs.

In Step S1603, CPSOCKET broadcasts its existence over the LAN interface via service advertising protocol broadcasts (SAPSERVER) which contain a proprietary socket number (see sectoin 4g above). Because other servers are operating in the multi-tasking environment established in Step S1602, and because the Netware®-compatible software only permits a single non-fileserver server to advertise from a single network node such as the NEB, CPSOCKET broadcasts its SAP advertisements via the SAPSERVER program. As described more fully above in paragraph 4g, the SAPSERVER program permits two network servers to broadcast from a single network node even when the network supports only single servers for each network node.

In Step S1604, CPSOCKET receives a broadcast request from a client, for example, CPINIT or CPCONSOL on proprietary socket 453. CPSOCKET responds to the client (Step S1605) with an IPX packet on the same socket.

In Step S1606, the client establishes direct SPX communication with CPSOCKET over a socket number that is pre-assigned to CPSOCKET, here socket number 83B4 for communication or 83B5 for connection. In accordance with that direct connection, CPSOCKET receives and interprets client requests and/or commands that are received over the LAN interface, monitors the status of the printer over the bi-directional SCSI interface, receives and sends status commands and/or inquiries to the printer via the bi-directional SCSI interface, reconfigures the NEB and the NEB configuration parameters, and sends requested information to the client via the LAN interface. These steps are described more fully below in connection with Steps S1607 through S1620 of FIGS. 16A and 16B.

In more detail, in Step S1607, if CPSOCKET determines that a configuration command has been received, then flow advances to Step S1608 in which the configuration commands are executed and the result provided via the LAN to the client. Configuration commands are listed in Table 9 and generally pertain to the configuration of the NEB board as either a CPSERVER or an CRPRINTER in accordance with configuration commands initiated by the CPINIT program.

Table 10: Device Information Commands

Command	Data (CPCONSOL → CPSOCKET)	Response (CPSOCKET → CPCONSOL)
request for interface status	none	interface status
request for control status	none	printer control information for CPCONSOL "control" menu
request for font status	none	printer font set
request for layout status	none	printer layout (portrait/landscape, etc.)
request for quality and common environment status	none	printer macros
request for duplex status	none	printer duplex mode
request for miscellaneous	none	miscellaneous printer info (collation, stapling, paper folding, paper trays, etc.)
request for default control status	none	default printer control information for CPCONSOL "control" menu
request for default font status	none	default printer font set
request for default layout status	none	default printer layout (portrait/landscape, etc.)
request for default quality and common environment status	none	default printer macros

Command	Data (CPCONSOL → CPSOCKET)	Response (CPSOCKET → CPCONSOL)
set default miscellaneous printer info	default miscellaneous printer info (collation, stapling, paper holding, paper trays, etc.)	confirmation

If in Step S1611, CPSOCKET determines that a configuration parameter command has been received, then flow advances to Step S1612 in which CPSOCKET executes the received command and provides the result via the LAN to the client. As shown in Table 11, configuration parameter commands pertain generally to parameter values stored in the NEB concerning time, date, safe printer environment information, logging options, log file size, etc.

Table 11

Configuration Parameter Commands		
Command	Data (CPINIT → CPSOCKET)	Response (CPSOCKET → CPINIT)
request for current configuration parameters	none	configuration parameters (e.g. time, data, safe printer environment info, logging options, etc.)
set new configuration parameters	configuration parameters (e.g. time, data, safe printer environment info, logging options, etc.)	confirmation

If in Step S1613 CPSOCKET determines that a NEB application program command has been received, then flow advances to Step S1614 in which CPSOCKET provides information on the current application program, namely RPRINTER, PSERVER, or LPR (for UNIX). Application program information generally includes server name, file server queue, device ID, etc., as detailed in Table 12.

Table 12

Application Program Information		
Command	Data (CPINIT → CPSOCKET)	Response (CPSOCKET → CPINIT)
request for CRPRINTER info	none	CRPRINTER info
set CRPRINTER info	new CRPRINTER info	confirmation
request for CPSERVER info	none	CPSERVER info
set CPSERVER info	new CPSERVER info	confirmation
request for CLPR info	none	CLPR info
set CLPR info	new CLPR info	confirmation

If in Step S1615 (FIG. 16B) CPSOCKET determined that a NEB/printer statistic command has been issued, then flow advances to Step S1616 in which CPSOCKET interrogates the printer through the bi-directional SCSI interface to obtain needed printer statistics. The statistics correspond to the network group displays described above in connection with CPCONSOL, as well as to print job statistics such as the total

then extract printer statistics and error events from the log file at any time. The network administrator's portion of such functions has been described above in paragraph 4i, and reference may be had to the discussion and tables set forth therein, especially Table 7 which indicates the content of the log file depending upon the logging level set by CPINIT.

As background, few LAN peripherals maintain their own statistics, but the NEB 2 includes the capability of logging the current status and daily statistics of printer 4 at midnight of each day. This relieves the system administrator from having to remember to do this on a daily basis. The status and statistics data may be stored in printer hard disk 114, printer NVRAM 111, NEB DRAM 220, or in the NEB NVRAM 228. The location of the stored log file may be selected by the network administrator depending upon the remaining memory capacity of each of those memories, and the statistics required by the logging level selected by the network administrator. For example, if the printer has a hard disk, the network administrator may choose the fairly-detailed "JOB", logging level so that voluminous statistics may be retained. On the other hand, if the printer has no hard disk, the network administrator may choose the less-detailed "ERROR" logging level so that less storage space is required. If the log file is filled, new error data will merely wrap around in the memory replacing old error data with new error data.

The NEB will automatically store printer statistics such as pages printed, jobs printed, off-line time, and print time each night for access for the system administrator at a later time. The statistics can be used to anticipate replacement of consumable printer supplies, such as toner, and to monitor user behavior such as leaving the printer off-line for extended periods of time.

In general, the logging function is accomplished by the printer controller board always knowing what time it is. When a printer/controller board is first powered on, the board finds the nearest server and requests the time. The board continues to do this every minute. When the day of the week changes, the board automatically requests the printer to report its page count. The board then calculates the daily statistics and stores them either to the printer hard disk or to the board NVRAM. These statistics are stored and available to the external network program CPCONSOL that can display them to a screen or save them to an external file.

As described above in paragraph 4i, the network administrator may select four logging levels: NONE; AUTO; ERROR; and JOB. At the NONE level, no logging statistics are maintained (although they may still be calculated every minute and temproarily kept in NEB DRAM 220). At the AUTO level, daily statistics are maintained for printer features such as printing days, pages, jobs, off-line time, and print time. The number of cumulative pages printed is determined by the printer, but the other statistics are determined by the NEB.

The ERROR logging level maintains the daily statistics discussed above, and also error conditions in the printer and also errors that occur in an application (i.e. CPSERVER). The NEB queries the printer every minute for such error conditions. Such printer error conditions may include: off-line; out-of-paper; printer-is-open; paper-jam; no-toner-cartridge; toner-is-low; printer feed and load errors; tray-is-full; line errors; print-job-rejected; font-is-full; service call; etc. Application errors may include: fileserver down; primary fileserver unavailable; CPSERVER running elsewhere; IPX not installed; etc.

The JOB logging level maintains the daily statistics and error conditions noted above and also maintains job start and job end information, which are determined by the NEB. Of course, the number and types of logging levels, and the data retained in each logging level may be varied according to the particular peripheral and the particular LAN in which the NEB is installed.

FIGS. 17A and 17B comprise a flow chart showing the overall operation of the automatic logging function within the NEB. Reference may also be had to FIG. 5A and Table 7 noted above. At Step S1, power is applied to the NEB and at Step S8, the timer module finds the nearest server and requests the time. At Step S1701, it is determined whether the NONE logging level has been selected. If the NONE logging level has been selected, the process skips to the end of the flowchart where a return is made to the overall flow diagram of FIGS. 5A, 5B and 5C.

If the NONE logging level has not been chosen in Step S1701, Step S1702 determines whether the AUTO logging level has been selected. If the AUTO logging level has been selected, the process proceeds to Step S9 where midnight is awaited. However, if the AUTO logging level has not been selected, Step S1703 determines whether the ERROR logging level has been selected. Where the ERROR logging level has been selected, the process skips to Step S1706 where a one minute timeout is awaited. However, if the ERROR logging level has not been selected, it is determined in Step S1704 that the JOB logging level has been selected. In this case, Step S1705 stores the job start and job end times to the log file. At Step S1706, a one minute timeout is awaited whereafter Step S1707 queries the printer for error events and saves such events to the log file. Thus, when either the ERROR or JOB logging levels have been selected, the board queries the printer every minute for error events and stores such error events in the log file.

Thus, the non-preemptive multi-tasking allocation of the microprocessor resources allows processing of a number of tasks in parallel on a near real-time basis.

4m. Placing The Printer In A Default Configuration

As discussed above with respect to Step S25 in FIG. 5C, the NEB will ensure that the printer is set to a known, default configuration at the beginning or end of a print job. The NEB does this by downloading to the printer's non-volatile memory (either hard disk 114 or NVRAM 111) a default configuration code which indicates the default environment (e.g. portrait mode, 10 point type, Roman lettering, etc.) in which the printer should be left at the conclusion of a print job. Upon receiving a print data stream from the LAN, the NEB retrieves the configuration code from the printer's non-volatile memory, appends the configuration code to a block of print data as an escape sequence, and then downloads the print job block with appended escape sequence to the printer. The printer will then conduct a printing operation, and (based on the escape sequence) will leave the printer in the desired default configuration.

Novell NetWare® software includes the ability to reset a network printer in a default environment after every print job. It does this by having the file server 30 install what amounts to a fake print job at the head of the print job itself. However, the exact printer escape sequences necessary to set particular printer default configurations reside in a database on the network, and not within the printer itself. Therefore, if it is desired to operate UNIX on the LAN, or where there is a problem with the file server itself, the printer may not be restored to a default configuration which ensures that the next print job will be printed with the printer in a known configuration.

A method of guaranteeing a printer default environment using the NEB operates on the difference that the printer reset state configuration and requisite escape sequence instructions reside within the printer itself, and the printer itself is responsible for resetting its own environment within print jobs. Thus, the printer reset feature is available without depending upon any device external to the printer. Furthermore, the initial default configuration may be loaded and subsequently modified from a remote location over the LAN through the NEB's serial or parallel interfaces.

The configuration code may be sent to the NEB through the CPCONSOL program, as discussed above in section 4i.

It may be convenient to store a plurality of default configuration codes in the printer non-volatile memory in order to allow the network administrator great flexibility for printer usage on the LAN. For example, print jobs received from an engineering source may require the printer to default to a portrait mode, whereas print jobs received from accounting may require that the printer be left in a spread sheet mode. Thus, by ensuring a known default environment, any of a number of LAN sources may utilize the printer for their specific jobs.

FIG. 19 depicts a more detailed flowchart for setting the printer default configuration. At Step S1, power is applied to the NEB, and at Step S22, the NEB accesses the LAN file server for active print queues and downloads print data to the DRAM 220.

If the printer non-volatile memory stores more than one default configuration code, it may be necessary to first determine what type of data is being transmitted from the LAN in order to determine which default configuration the printer should be left in. Therefore, Step S9101 determines the LAN source of the print job, and Step S1902 retrieves the appropriate default configuration code from the printer, which code corresponds to the determined LAN source.

At Step S1903, the NEB assembles blocks of image data and designates a start-of-print-job and an end-of-print-job for each print job. At Step S1904, the NEB microprocessor 216 appends to a print job an escape sequence which corresponds to the retrieved configuration code. Preferably, the escape sequence is appended to the beginning of the print job, but it may be appended to the end of the print job, or to both the beginning and end of the job. Then, at Step S1905, the print job, with appended escape sequence, is transferred to the printer, and the printer then renders print according to the received print job. When a print job is completed after Step S24, the printer will set itself to a default environment at Step S25, which environment corresponds to the default configuration code retrieved in Step S1902. Therefore, the printer will be left in a default environment which ensures that the next print job will begin with the printer in a known configuration.

Thus, a robust and efficient hardware and software solution has been found for ensuring that the printer itself stores a default configuration and is responsible for placing itself in a default condition at the end of every print job.

S2104, it is determined whether the last module identified by the configuration file has been selected. If the last module has not been selected, the process loops to Step S2103, where the header is written for the next module.

When the last module has been selected in Step S2104, the utility appends the ROM-resident code to the end of the image program (at Step S2105) so that upon power-up, the initialization code resides at the address expected by microprocessor 216.

When the ROM binary image is thus constructed, the image may be downloaded to one portion of the memory area of NEB DRAM 220, and then flashed to EPROM 222, as will be discussed in greater detail in section 4q below and with respect to the detailed discussion of FIG. 5C, Step S36.

4p. Protecting The EPROM During A Flash Operation

FIG. 22 is a block diagram showing the functional construction of the EPROM flash protection circuitry resident on the NEB. The EPROM flash protection circuit includes microprocessor 216 coupled to data bus 250 and address bus 251. Also connected to data bus 250 and address bus 251 is DRAM 220. DRAM 220 is capable of storing a ROM firmware image downloaded from a remote LAN device into one portion of its memory area (see section 4o above), and application process steps into another portion of its memory area. Also coupled to data bus 250 and address bus 251 are EPROM 222, latch 252, and PAL 253. D-type flip-flop 254 is connected to latch 252 and PAL 253. During operation, flip-flop 254 receives as its clock input an output signal from PAL 253 and as its data input, an output signal from latch 252. Latch 252 and PAL 253 are also connected to DC-DC converter 212, and DC-DC converter 212 is connected to transistor switch 255. When activated by latch 252, DC-DC converter 212 sends +12 volts to the input emitter of transistor switch 255. Flip-flop 254 is also connected to transistor switch 255 to provide the necessary input to open/close switch 255.

The operation of the EPROM flash protect circuitry will now be explained in more detail with reference to FIG. 22. Upon power-up, output of latch 252 will be low and flip-flop 254 will be reset. In this manner, the output signal PROG1 from latch 252 will be low and voltage from DC-DC converter 212 will be directed to sink current to a ground state. At power-up, flip-flop 254 is reset so that its output is set low thereby opening transistor switch 255.

With transistor switch 255 in an open state, Vpp pin of EPROM 222 will be held at 0 volts preventing any data from being accepted or a flash operation from being performed. That is, for a flash operation to occur in EPROM 222, the Vpp pin must reach a level of at least +11.4 volts, which is a requirement set by the EPROM manufacturer's specifications. However, in order to achieve this voltage level, the following two programming steps are required.

First, when a new ROM firmware package is received in DRAM 220, microprocessor 216 receives a command to flash EPROM 222, by generating an I/O write to address 360 hex with data bit 7 high (80 hex). In this manner, DC-DC converter 212 can be first turned on.

As shown in Tables 16 and 17, address 360 hex corresponds to control register 230 which is used to control read/write operations to NVRAM 228. As shown in Table 17 below, when 360 hex is sent with bit 7 high/low, the address corresponds to an operation of DC-DC converter 212.

firmware image.

The operation of the EPROM protection circuit will now be explained with reference to FIG. 22 and the flowchart of FIG. 23.

In Step S2301, a new ROM firmware image is received by NEB 2 across the LAN and loaded into DRAM 220. Microprocessor 216 receives a command to flash EPROM 222 in Step S2302. In Step S2303, microprocessor 216 sends out an I/O write command to PAL 253 and outputs address 360 hex with bit 7 high. Flow advances to Step S2304 in which bit 7 high activates latch 252 to output the PROG1 signal. The PROG1 signal turns on DC-DC converter 212 and +12 volts is output to transitor switch 255. In Step S2305, microprocessor 216 sends both an I/O read command to PAL 253 and address 366 which is a PAL address. In response, PAL 253 outputs the PROG2 signal to clock flip-flop 254 which allows the PROG1 signal to be input at its data input. Flip-flop 254 outputs the TRANSON signal to transistor switch 255 which allows +12 volts to pass from the collector of transistor switch 255 to the Vpp pin of EPROM 222. In Step S2306, microprocessor 216 clears and then erases EPROM 222. In Step S2307, microprocessor 216 determines if EPROM 222 has been completely erased. If EPROM 222 is not completely erased, flow returns to Step S2307.

After microprocessor 216 determines that EPROM 222 has been completely erased, in Step S2308, the ROM firmware image is downloaded from DRAM 220 to EPROM 222. Once the ROM firmware image is successfully loaded, in Step S2309 microprocessor 216 writes address 360 hex with bit 7 low. The PROG1 signal from latch 252 goes low and DC-DC converter 212 allows the voltage level to sink current to a ground state.

In Step S2310, microprocessor 216 sends PAL 253 an I/O read command and a 366 hex address which permits the PROG2 signal to go low thereby clocking the flip-flop which outputs a low TRANSON signal which operates to open transistor switch 255.

Thus, in Steps S2309 and S2310, +12 volts is removed from Vpp pin of EPROM 222 and the flash operation is ended. After the flash operation, microprocessor 216 determines if a re-boot command has been received in Step S2311. If the re-boot command has been received, NEB 2 is re-booted in Step S2312 from the new ROM firmware image in EPROM 222. However, if no re-boot command is received, then flow ends.

4q. Remotely Altering Firmware

The method for remotely altering firmware in EPROM 222 will be discussed in more detail below and with reference to the flowchart illustrated in FIG. 24, Step S36 of FIG. 5C, and section 4i above.

Prior to shipping a NEB to a customer, the NEB is configured with the minimum number of executable files which permit the NEB to perform necessary functions. However, the NEB can be reconfigured subsequently by the customer. That is, a network administrator may download data from a remote LAN device, which data may contain anything from a patch code, to manufacturing test routines, to entire firmware updates to be downloaded to the EPROM.

In more detail, NEB2 can be reconfigured by sending executable files across the LAN from the network administrator's PC 14 to NEB 2. The network administrator can remotely alter the ROM firmware image in EPROM 222, as desired.

In Step S2401, the network administrator activates a CPFLASH program that uses a MAC address as a command line parameter to target a specific NEB. CPFLASH issues a SAP broadcast request which is responded to by SAPSERVER running on the NEB. In Step S2402, CPFLASH waits for a response from the targeted NEB. If in the case where the targeted NEB does not respond in approximately 15 seconds, the flow returns to Step S2401 and the broadcast is resent. However, in the case where the targeted server responds, flow advances to Step S2403.

In Step S2403, the address and location of the targeted NEB is received, communication with the NEB having the matching MAC address is established, and a new ROM image firmware is downloaded over the LAN to DRAM 220.

In Step S2404, the validity of the ROM firmware image is checked before proceeding to the next step. The validity of the ROM firmware image is verified against an image checksum which is sent in a special packet along with the download operation in Step S2403. If the checksum value does not match the checksum downloaded with the ROM image, then in Step S2405 the operator is notified of an error and the ROM firmware image in DRAM 220 is purged.

If the checksum value is valid, then flow advances to Step S2406 at which point microprocessor 216 retrieves any data which is to be preserved, such as the MAC address, and stores the data within the proper locations in the new firmware image stored in DRAM 220. In this fashion, if the new ROM firmware

At Step S2605, the POST program is complete and NEB 2 waits for instructions from across any one of the ports, preferably the serial port. The waiting period can be approximately a one second window in which time PC1 300 should respond with the prepared text programs. In Step S2606, if PC1 300 does not respond by sending a test program to NEB 2 within the time window, flow advances to Step S2607 where the NEB enteres its normal operational mode.

When the test program instruction set from PC1 300 is received in Step S2606, the instruction set, which includes further test programs, is stored (in Step S2608) on NEB 2 in DRAM 220. In Step S2609, PC1 300 activates the instruction set and NEB 2 executes each test program within the instruction set.

The test program instruction set may contain, in random order, test programs which require NEB 2 to configure PC2 306 as a LAN peripheral device, or which require NEB 2 to configure PC2 306 as an SCSI peripheral device. In either case, after being configured, PC2 306 will respond to each communication from NEB 2, usually by merely returning data blocks sent by the NEB.

Briefly, in Step S2610 (FIG. 26B) NEB 2 configures PC2 306 as a LAN peripheral and PC2 306 responds by sending a response to NEB 2 which effectively performs a LAN loopback test by returning the data which it has received. NEB 2 will communicate with PC2 and receive simulated print job results. In Step S2611, the result of each block job is sent to PC1 300. PC1 300 determines if the test result is correct. In Step S2611, if it is determined by PC1 300 that the test result is incorrect, PC1 300 sends a re-scripted, branch test program (Step S2612) in accordance with the test result received in Step S2611. However, if no further branch test program exists, then in Step S2612 PC1 300 will stop LAN testing and output an error signal.

Thus, in Step S2611, NEB 2 is tested for LAN communications. Assuming NEB 2 successfully passes each LAN communication test, flow advances to Step S2613 at which point PC2 306 is configured as an SCSI peripheral device and performs SCSI loopback tests by returning the data which it has received. In Step S2614 the results of the tests are sent to PC1 300 and if the results are incorrect, PC1 300 similarly sends a branch test in Step S2615 in accordance with the test result. Of course, if no further branch test exists to further test the peripheral communication, then PC1 300 stops the test, and outputs an error signal.

Assuming that NEB 2 successfully passes each SCSI communication test in Step S2614, then flow advances to Step S2616 at which point NEB 2 requests further instructions from PC1 300. If PC1 300 returns with further instructions, flow returns to Step S2605, but if further testing is not necessary then NEB testing is ended.

In summary, a method for testing an interactive network board having a LAN interface and a test interface comprises the steps of applying power to the board and reading a POST result which was executed out of board ROM via the test interface, and downloading a test program into the board RAM via the test interface. The test program is then activated for execution out of board RAM. The board may then be commanded to configure a peripheral device (through either the LAN or the SCSI interface) to be a LAN driver or an SCSI peripheral. The board then interacts with the LAN driver or SCSI peripheral in accordance with the test program. Results of the test program are then output via the test interface to a test computer which receives these test results. If certain tests fail, additional test programs may be scripted in accordance with the type of failure. The newly scripted test programs will be able to perform fault detection and diagnosis, and these additionally scripted test programs may then be downloaded to the board RAM from the PC1.

Once all of the tests are successfully concluded, it may be convenient (in the factory test environment) to flash the operational firmware into EPROM 222. Specifically, the last step of a testing program may be utilized to load the requisite firmware image into the NEB EPROM 222 prior to delivery (see section 4q above). The firmware flashed to EPROM 222 may also include a unique MAC address for NEB 2.

In the past, MAC addresses were incorporated into circuit boards using a dedicated PROM chip such as PROM 232. However, it has been found that if the MAC address is flashed into EPROM, the PROM chip is not required, while the MAC address can still be stored in a non-volatile way. (Of course, as discussed in paragraph 4q, the MAC address could also be remotely flashed into the EPROM at the same time the RAM firmware image is updated, after NEB 2 is coupled to the LAN.)

In Step S2617 of FIG. 26B, NEB testing has been completed and each board may be designated with its own individual identifier number, commonly referred to as a MAC address. Thus, in Step S2617 it is determined whetehr a ROM firmware image is to be stored in EPROM 222. If no image is to be stored, testing ends. However, if an image is to be stored, flow advances to Step S2618 where the ROM image (with MAC address) is flashed to EPROM 222. At Step S2618 it may also be desirable to download other data normally stored in PROM 232, such as board revision number, data of manufacture, tester name, etc., together with the MAC address.

modules into said RAM.

5. Apparatus according to Claim 4, wherein said LAN interface receives a new configuration file from the LAN, said new configuration file corresponding to the second subset of modules, and wherein said processor (1) stores the new configuration file in said NVRAM, (2) upon a next power up of said board, reads the new configuration file from said NVRAM, (3) selects from said PROM the second subset of modules in response to reading the new configuration file from said NVRAM, and (4) loads the selected second subset of modules into said RAM.

6. Apparatus according to Claim 1, wherein said processor executes said application module and said control module on a parallel, multi-tasking basis.

7. Apparatus according to Claim 1, wherein said peripheral interface comprises a bi-directional interface.

8. Apparatus according to Claim 7, wherein said bi-directional interface comprises an SCSI interface.

9. Apparatus according to Claim 8, further comprising an SCSI controller disposed on said board, for controlling said SCSI interface.

10. Apparatus according to Claim 1, wherein the peripheral comprises a printer, wherein said peripheral interface comprises an SCSI interface, and wherein said processor executes said application module and said control module simultaneously on a non-preemptive multi-tasking basis.

11. Apparatus according to Claim 1, wherein said RAM comprises a dynamic RAM.

12. Apparatus according to Claim 1, further comprising a serial port coupled to said board.

13. Apparatus according to Claim 1, wherein the peripheral comprises a printer, wherein said peripheral job information comprises a print job data stream received by said LAN interface, wherein said peripheral job data comprises a print job, and wherein said processor adds a beginning-of-job indication to a front of said print job and an end-of-job indication to an end of said print job.

14. Apparatus according to Claim 1, wherein said LAN interface receives a further application module from the LAN, and wherein said processor causes said further applications module to be stored in said PROM.

15. A circuit board for coupling a printer to a LAN, comprising:

a printer interface for transmitting print data to the printer, and for receiving printer status data from the printer;

a LAN interface for receiving print job information and printer status requests from the LAN, and for transmitting printer status information to the LAN;

a ROM for storing a first program for placing said board in a first operational configuration, and a second program for placing said board in a second operational configuration different from said first configuration; and

a processor for executing one of the first and second programs.

16. A board according to Claim 15, wherein said first program places the board in a PSERVER operational configuration, and wherein said second program places said board in an RPRINTER operational configuration.

17. A board according to Claim 15, further comprising a RAM for temporarily storing the first or second program during execution thereof.

18. A board according to Claim 15, wherein said printer interface comprises an SCSI interface.

19. A board according to Claim 15, wherein said ROM also stores a status and control program for processing the printer status data, the printer status requests, and the printer status information, and wherein said processor executes said one of said first and second programs and said status and

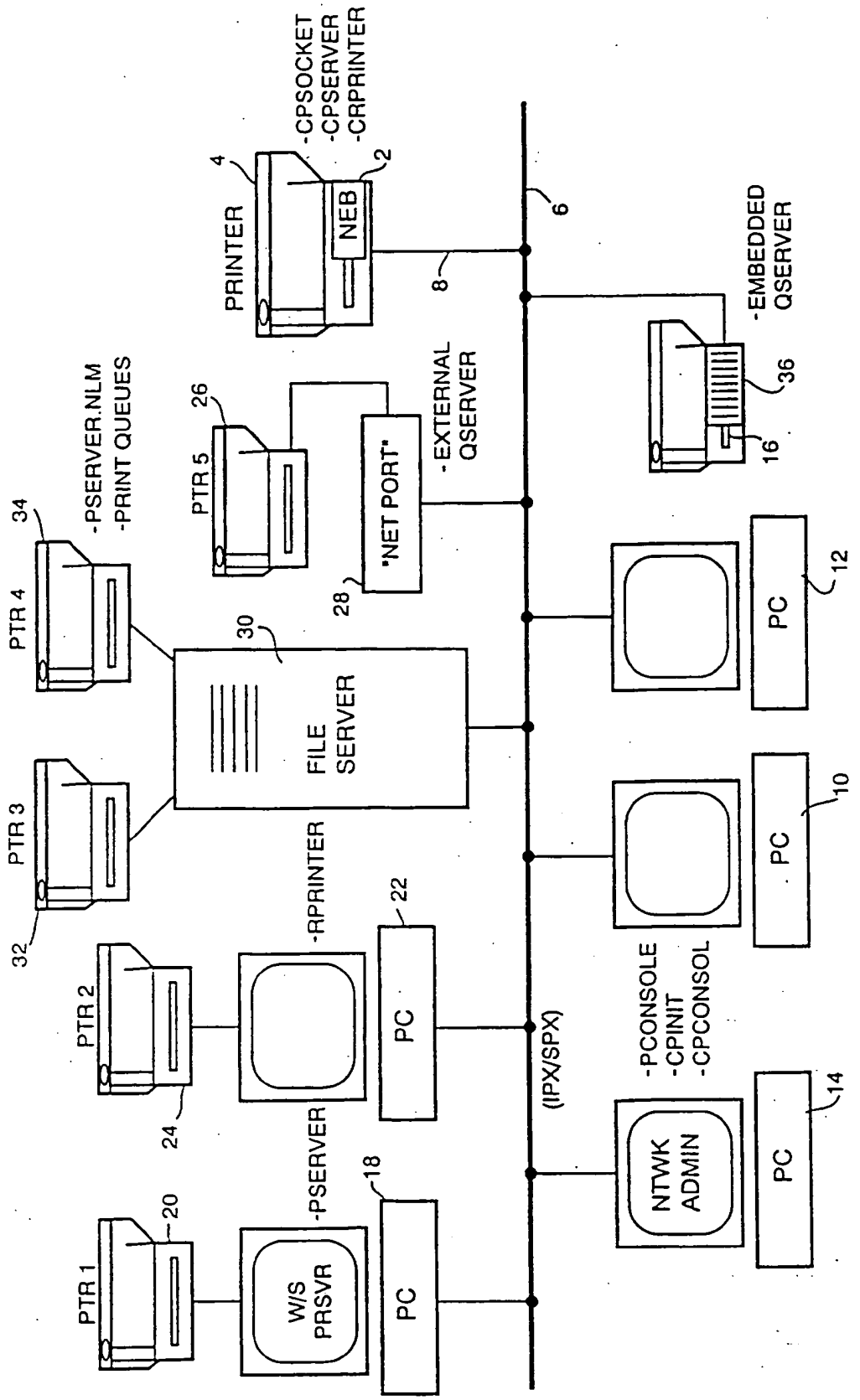
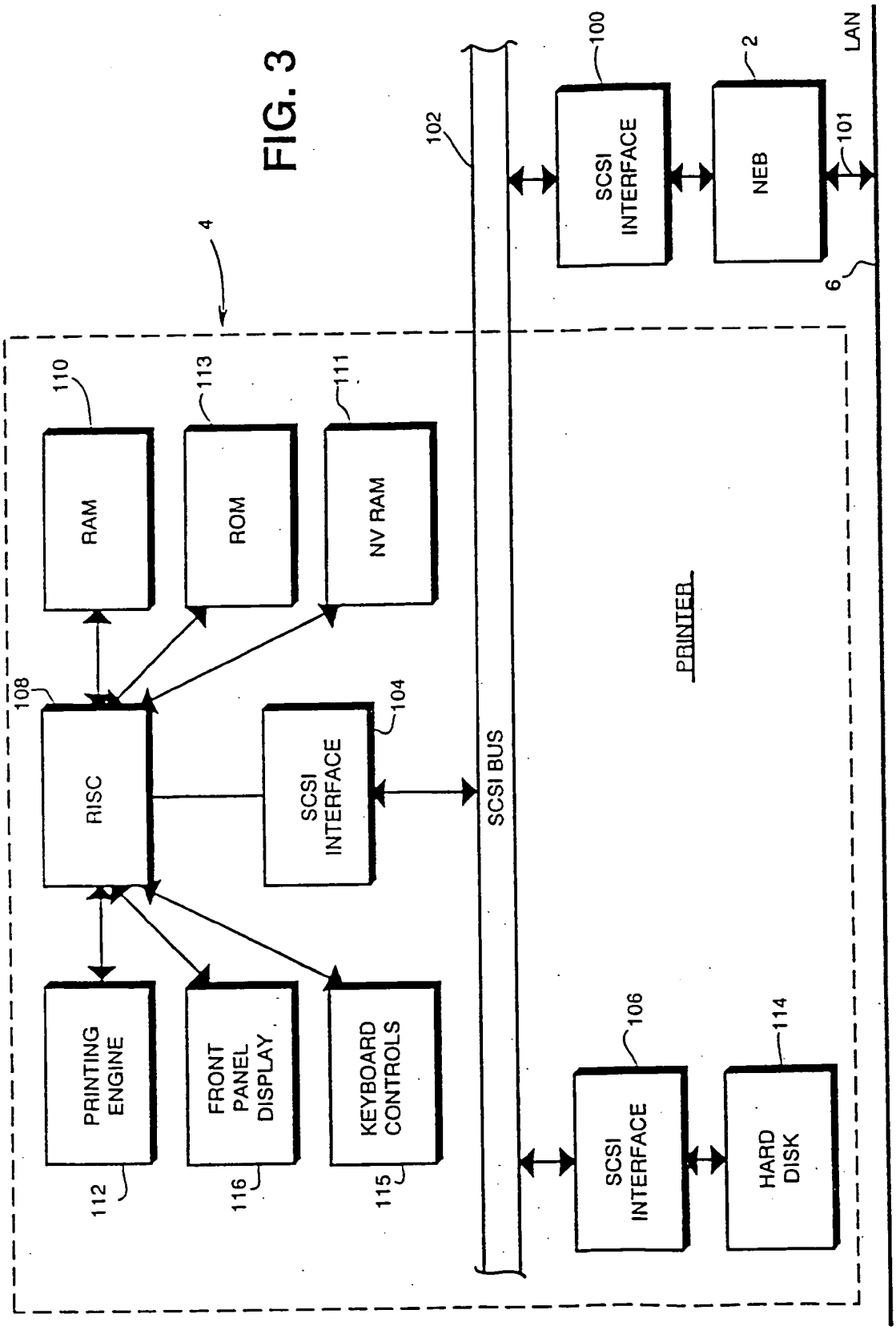

PRINTER
4
-CPSOCKET
-CPSERVER
-CRPRINTER
NEB
2
6
8
-EMBEDDED
QSERVER
36
16
PTR 5
26
"NET PORT"
28
-EXTERNAL
QSERVER
34
-PSERVER.NLM
-PRINT QUEUES
PTR 4
30
FILE
SERVER
PTR 3
32
PC
12
PC
10
PTR 2
24
-RPRINTER
22
PC
(IPX/SPX)
-PCONSOLE
-CPINIT
-CPCONSOL
NTWK
ADMIN
PC
14
PTR 1
20
-PSERVER
W/S
PRSVR
18
PC

FIG. 1

FIG. 3


LAN
100
2
SCSI INTERFACE
NEB
101
102
6
4
110
113
111
RAM
ROM
NV RAM
108
RISC
SCSI INTERFACE
104
SCSI BUS
PRINTER
PRINTING ENGINE
FRONT PANEL DISPLAY
KEYBOARD CONTROLS
112
116
115
106
114
SCSI INTERFACE
HARD DISK

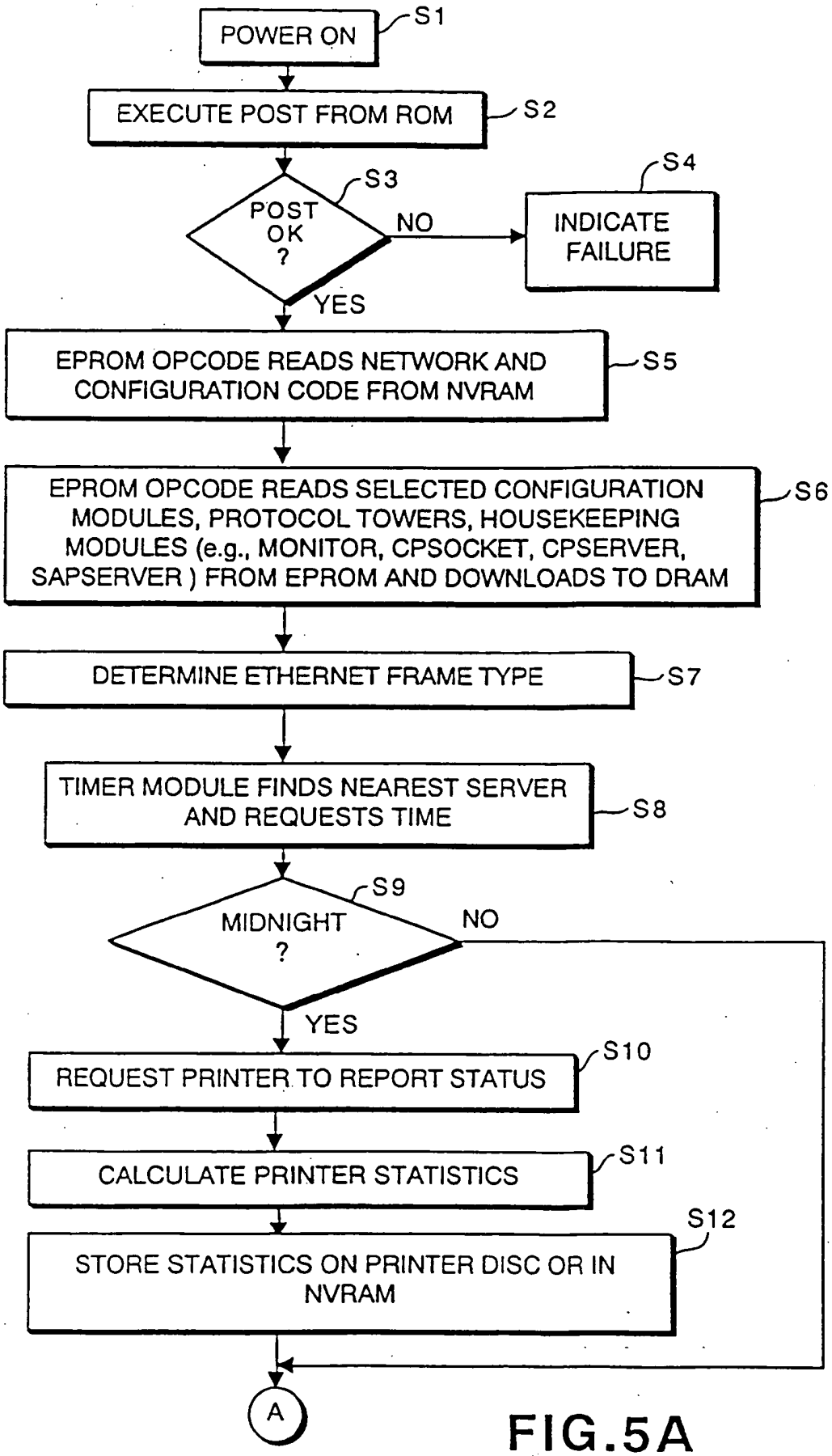

POWER ON
S1
EXECUTE POST FROM ROM
S2
POST OK ?
S3
NO
INDICATE FAILURE
S4
YES
EPROM OPCODE READS NETWORK AND CONFIGURATION CODE FROM NVRAM
S5
EPROM OPCODE READS SELECTED CONFIGURATION MODULES, PROTOCOL TOWERS, HOUSEKEEPING MODULES (e.g., MONITOR, CPSOCKET, CPSERVER, SAPSERVER) FROM EPROM AND DOWNLOADS TO DRAM
S6
DETERMINE ETHERNET FRAME TYPE
S7
TIMER MODULE FINDS NEAREST SERVER AND REQUESTS TIME
S8
MIDNIGHT ?
S9
NO
YES
REQUEST PRINTER TO REPORT STATUS
S10
CALCULATE PRINTER STATISTICS
S11
STORE STATISTICS ON PRINTER DISC OR IN NVRAM
S12
A

FIG.5A

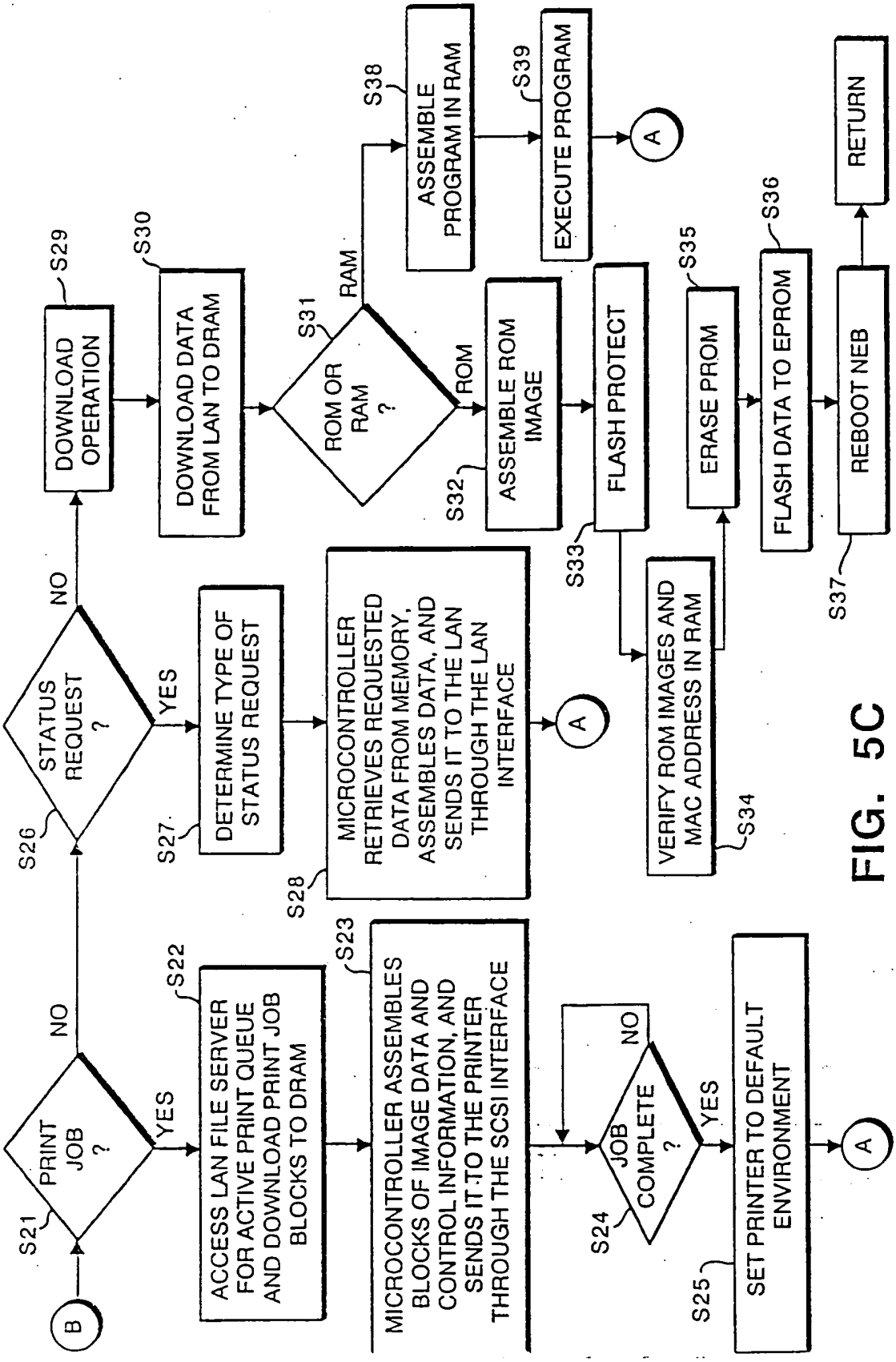

B
S21
PRINT JOB ?
NO
YES
S22
ACCESS LAN FILE SERVER FOR ACTIVE PRINT QUEUE AND DOWNLOAD PRINT JOB BLOCKS TO DRAM
S23
MICROCONTROLLER ASSEMBLES BLOCKS OF IMAGE DATA AND CONTROL INFORMATION, AND SENDS IT TO THE PRINTER THROUGH THE SCSI INTERFACE
S24
JOB COMPLETE ?
NO
YES
S25
SET PRINTER TO DEFAULT ENVIRONMENT
A
S26
STATUS REQUEST ?
NO
YES
S27
DETERMINE TYPE OF STATUS REQUEST
S28
MICROCONTROLLER RETRIEVES REQUESTED DATA FROM MEMORY, ASSEMBLES DATA, AND SENDS IT TO THE LAN THROUGH THE LAN INTERFACE
A
S29
DOWNLOAD OPERATION
S30
DOWNLOAD DATA FROM LAN TO DRAM
S31
ROM OR RAM ?
ROM
RAM
S32
ASSEMBLE ROM IMAGE
S33
FLASH PROTECT
S34
VERIFY ROM IMAGES AND MAC ADDRESS IN RAM
S35
ERASE PROM
S36
FLASH DATA TO EPROM
S37
REBOOT NEB
S38
ASSEMBLE PROGRAM IN RAM
S39
EXECUTE PROGRAM
A
A
RETURN

FIG. 5C

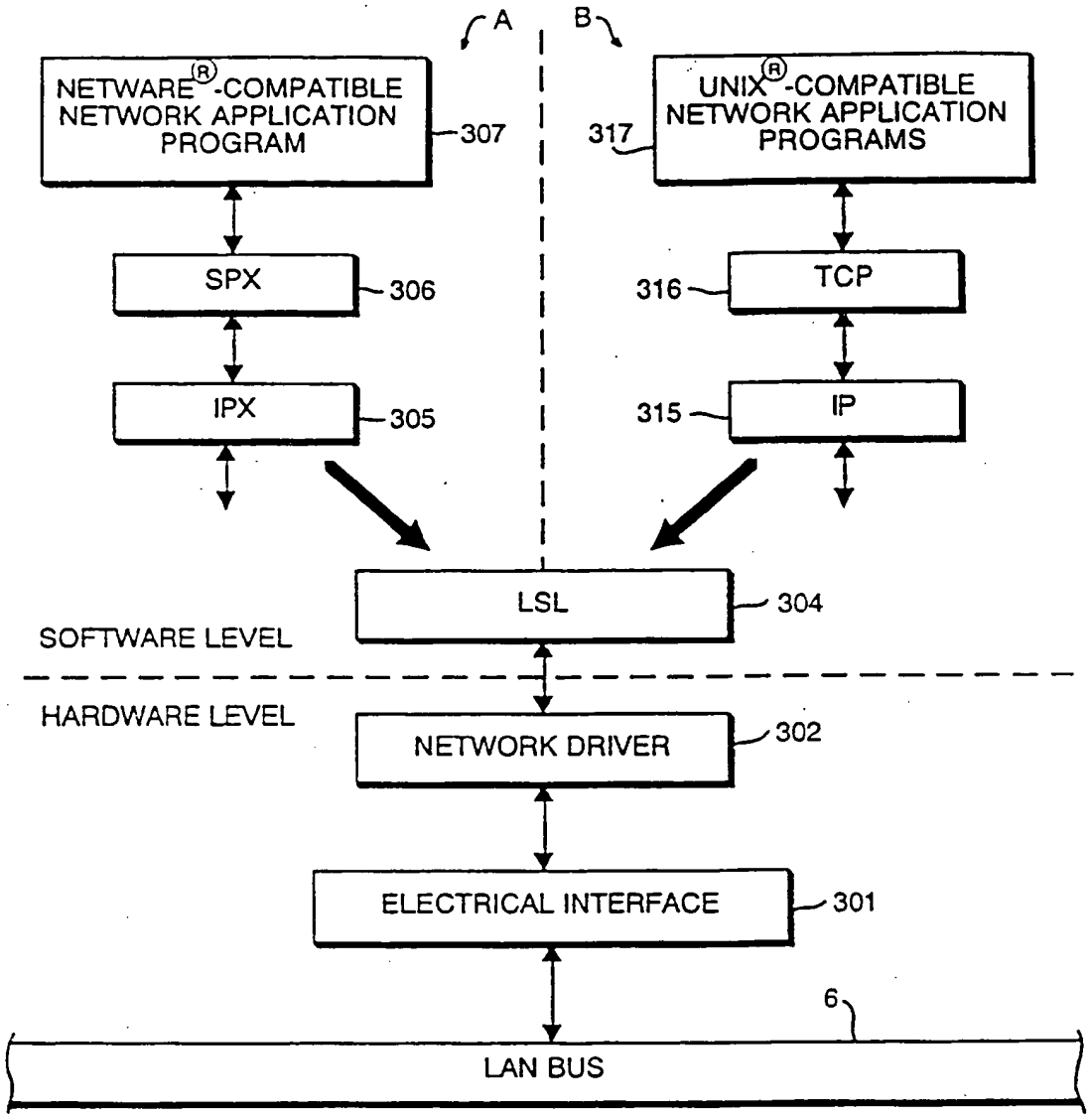

A
B
NETWARE®-COMPATIBLE NETWORK APPLICATION PROGRAM
307
UNIX®-COMPATIBLE NETWORK APPLICATION PROGRAMS
317
SPX
306
316
TCP
IPX
305
315
IP
LSL
304
SOFTWARE LEVEL
HARDWARE LEVEL
NETWORK DRIVER
302
ELECTRICAL INTERFACE
301
6
LAN BUS

FIG.7

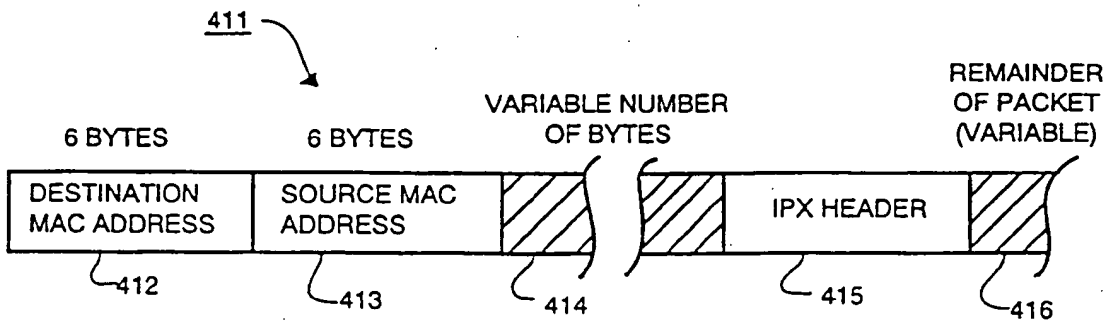

411
6 BYTES
6 BYTES
VARIABLE NUMBER OF BYTES
REMAINDER OF PACKET (VARIABLE)
DESTINATION MAC ADDRESS
SOURCE MAC ADDRESS
IPX HEADER
412
413
414
415
416

FIG. 9

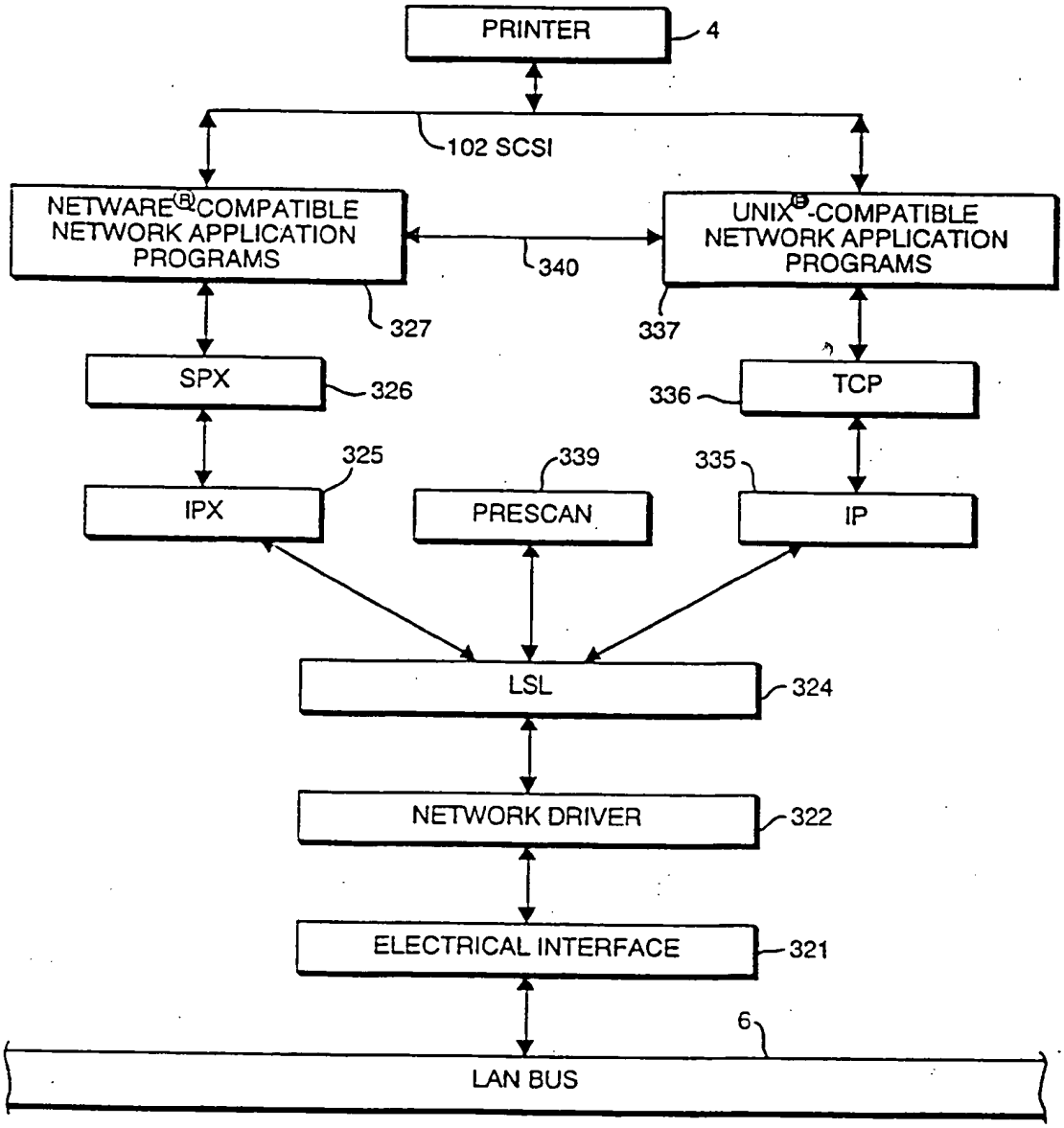

PRINTER
4
102 SCSI
NETWARE® COMPATIBLE NETWORK APPLICATION PROGRAMS
340
UNIX® -COMPATIBLE NETWORK APPLICATION PROGRAMS
327
337
SPX
326
336
TCP
325
339
335
IPX
PRESCAN
IP
LSL
324
NETWORK DRIVER
322
ELECTRICAL INTERFACE
321
6
LAN BUS

FIG.11

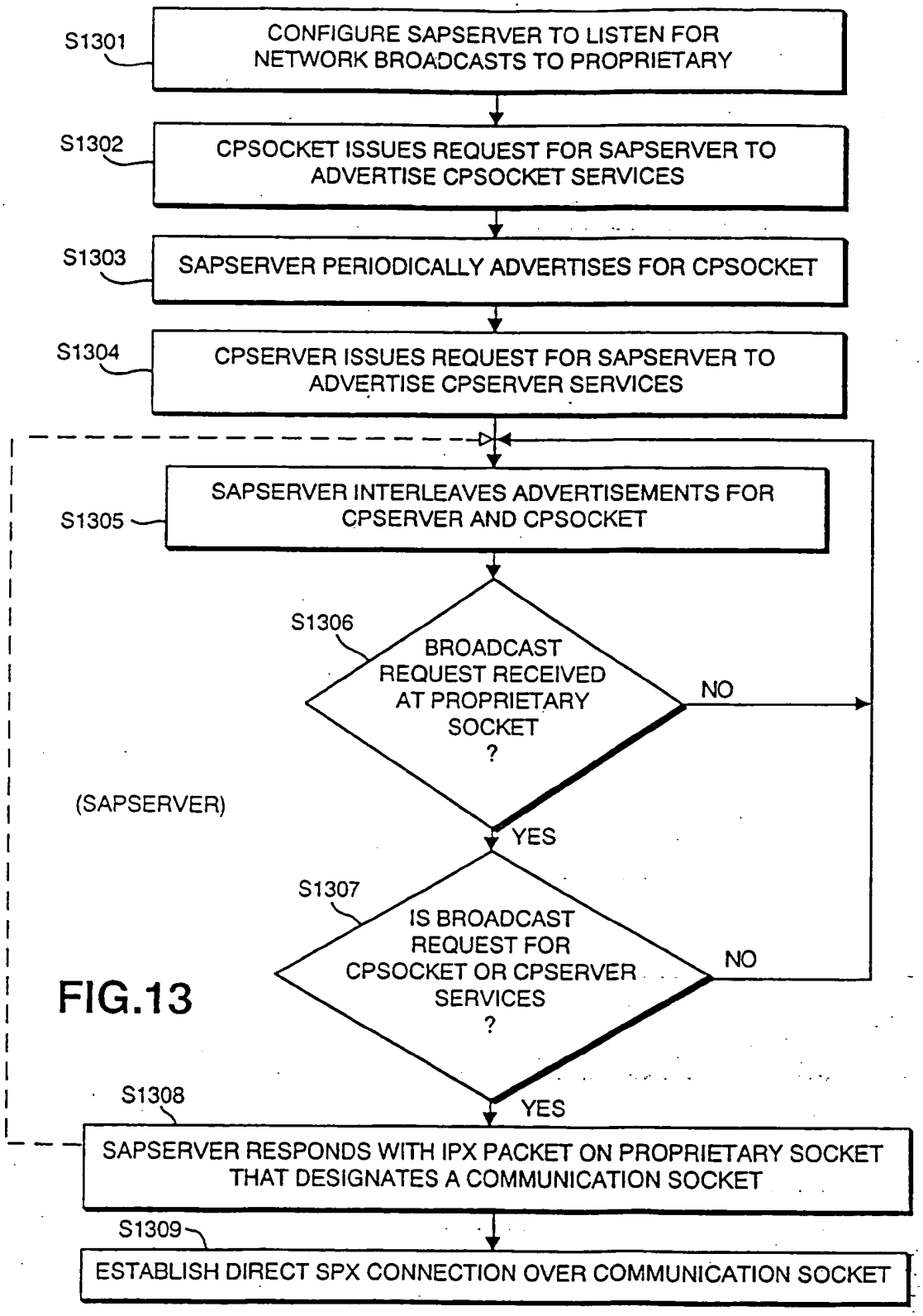

S1301
CONFIGURE SAPSERVER TO LISTEN FOR NETWORK BROADCASTS TO PROPRIETARY
S1302
CPSOCKET ISSUES REQUEST FOR SAPSERVER TO ADVERTISE CPSOCKET SERVICES
S1303
SAPSERVER PERIODICALLY ADVERTISES FOR CPSOCKET
S1304
CPSERVER ISSUES REQUEST FOR SAPSERVER TO ADVERTISE CPSERVER SERVICES
S1305
SAPSERVER INTERLEAVES ADVERTISEMENTS FOR CPSERVER AND CPSOCKET
S1306
BROADCAST REQUEST RECEIVED AT PROPRIETARY SOCKET ?
NO
YES
(SAPSERVER)
S1307
IS BROADCAST REQUEST FOR CPSOCKET OR CPSERVER SERVICES ?
NO
YES
S1308
SAPSERVER RESPONDS WITH IPX PACKET ON PROPRIETARY SOCKET THAT DESIGNATES A COMMUNICATION SOCKET
S1309
ESTABLISH DIRECT SPX CONNECTION OVER COMMUNICATION SOCKET

FIG.13

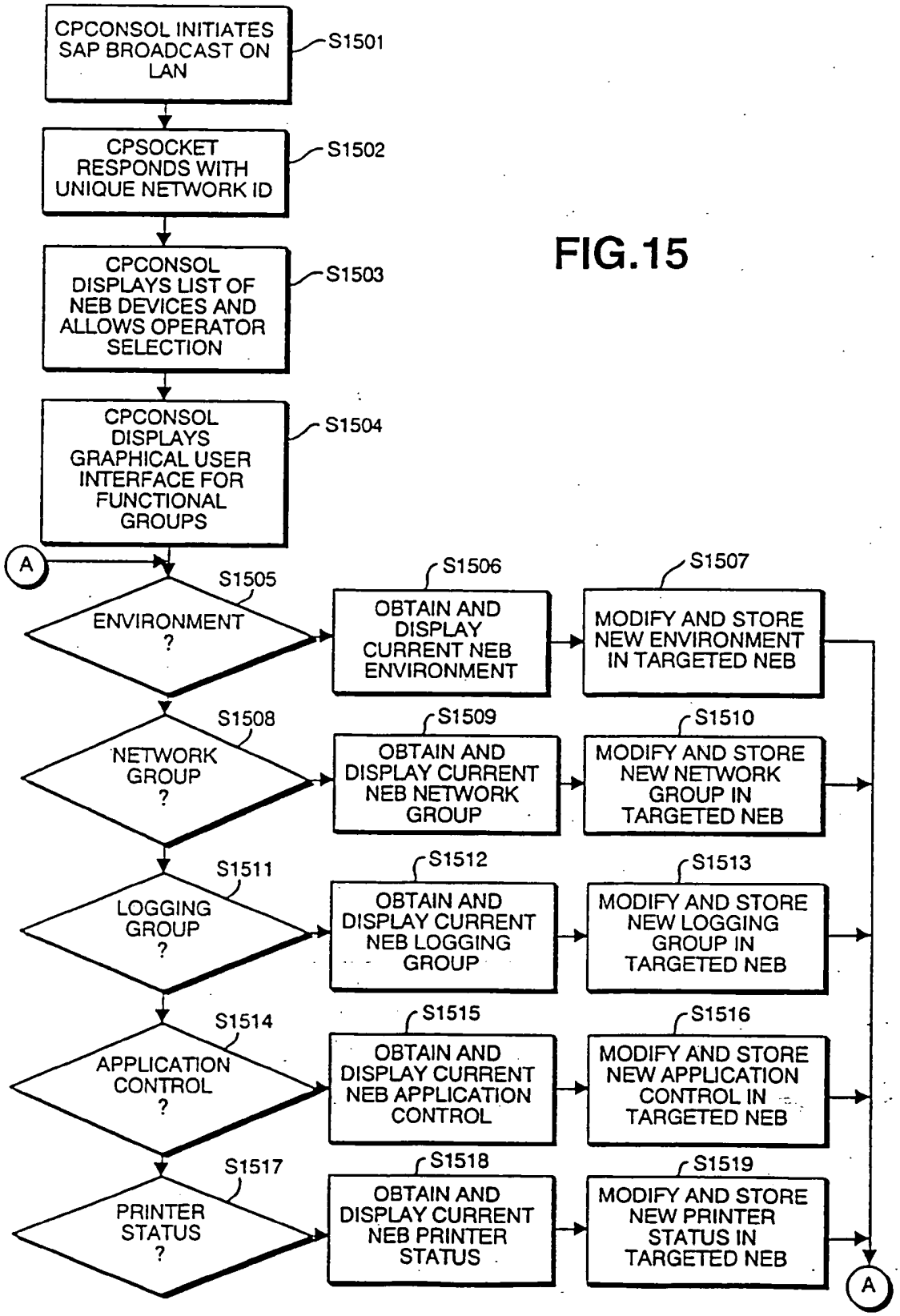

CPCONSOL INITIATES SAP BROADCAST ON LAN
S1501
CPSOCKET RESPONDS WITH UNIQUE NETWORK ID
S1502
CPCONSOL DISPLAYS LIST OF NEB DEVICES AND ALLOWS OPERATOR SELECTION
S1503
CPCONSOL DISPLAYS GRAPHICAL USER INTERFACE FOR FUNCTIONAL GROUPS
S1504
A
S1505
ENVIRONMENT ?
S1506
OBTAIN AND DISPLAY CURRENT NEB ENVIRONMENT
S1507
MODIFY AND STORE NEW ENVIRONMENT IN TARGETED NEB
S1508
NETWORK GROUP ?
S1509
OBTAIN AND DISPLAY CURRENT NEB NETWORK GROUP
S1510
MODIFY AND STORE NEW NETWORK GROUP IN TARGETED NEB
S1511
LOGGING GROUP ?
S1512
OBTAIN AND DISPLAY CURRENT NEB LOGGING GROUP
S1513
MODIFY AND STORE NEW LOGGING GROUP IN TARGETED NEB
S1514
APPLICATION CONTROL ?
S1515
OBTAIN AND DISPLAY CURRENT NEB APPLICATION CONTROL
S1516
MODIFY AND STORE NEW APPLICATION CONTROL IN TARGETED NEB
S1517
PRINTER STATUS ?
S1518
OBTAIN AND DISPLAY CURRENT NEB PRINTER STATUS
S1519
MODIFY AND STORE NEW PRINTER STATUS IN TARGETED NEB
A

FIG.15

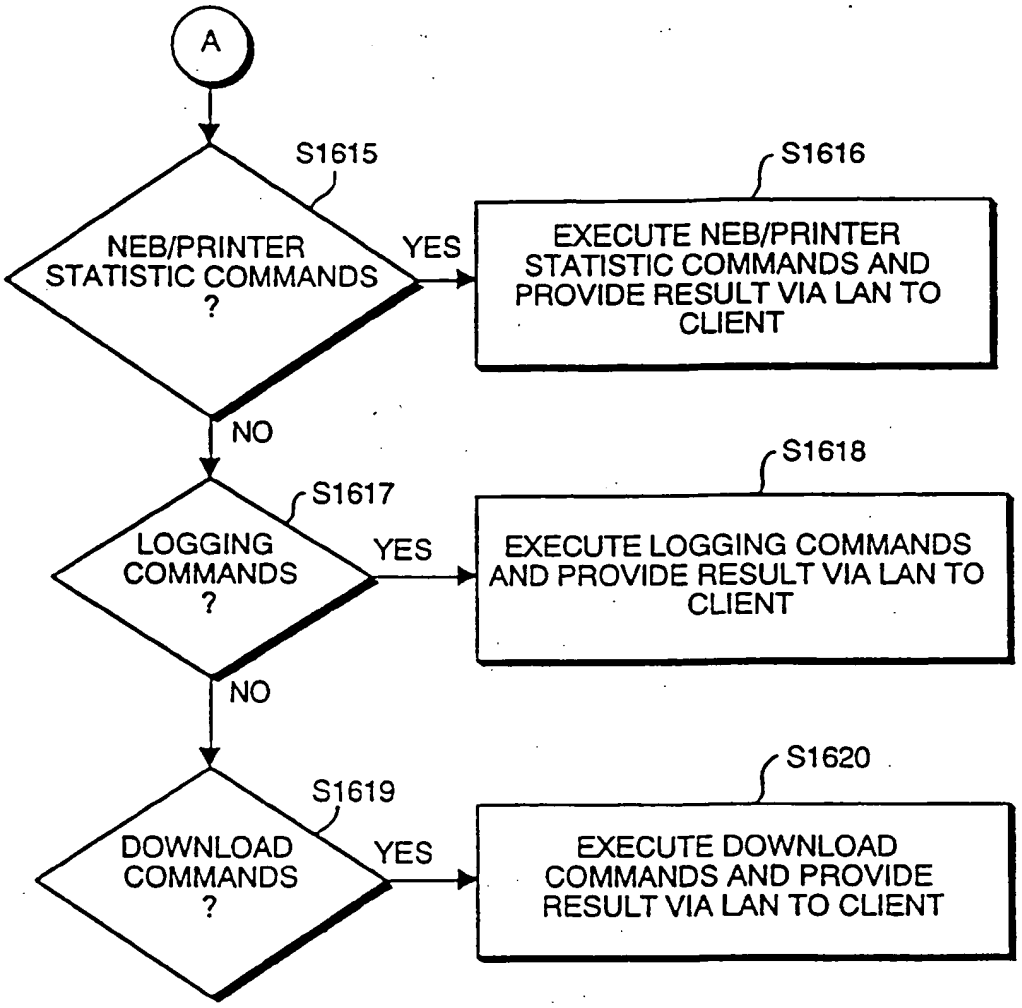

A
S1615
NEB/PRINTER STATISTIC COMMANDS ?
YES
S1616
EXECUTE NEB/PRINTER STATISTIC COMMANDS AND PROVIDE RESULT VIA LAN TO CLIENT
NO
S1617
LOGGING COMMANDS ?
YES
S1618
EXECUTE LOGGING COMMANDS AND PROVIDE RESULT VIA LAN TO CLIENT
NO
S1619
DOWNLOAD COMMANDS ?
YES
S1620
EXECUTE DOWNLOAD COMMANDS AND PROVIDE RESULT VIA LAN TO CLIENT

FIG.16B

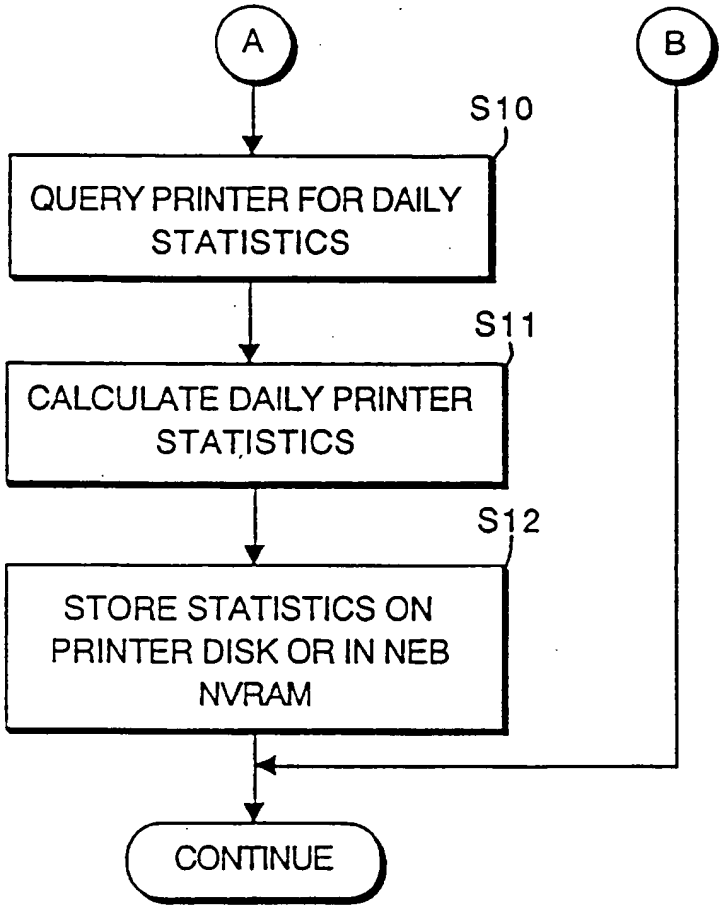

A
B
S10
QUERY PRINTER FOR DAILY STATISTICS
S11
CALCULATE DAILY PRINTER STATISTICS
S12
STORE STATISTICS ON PRINTER DISK OR IN NEB NVRAM
CONTINUE

FIG.17B

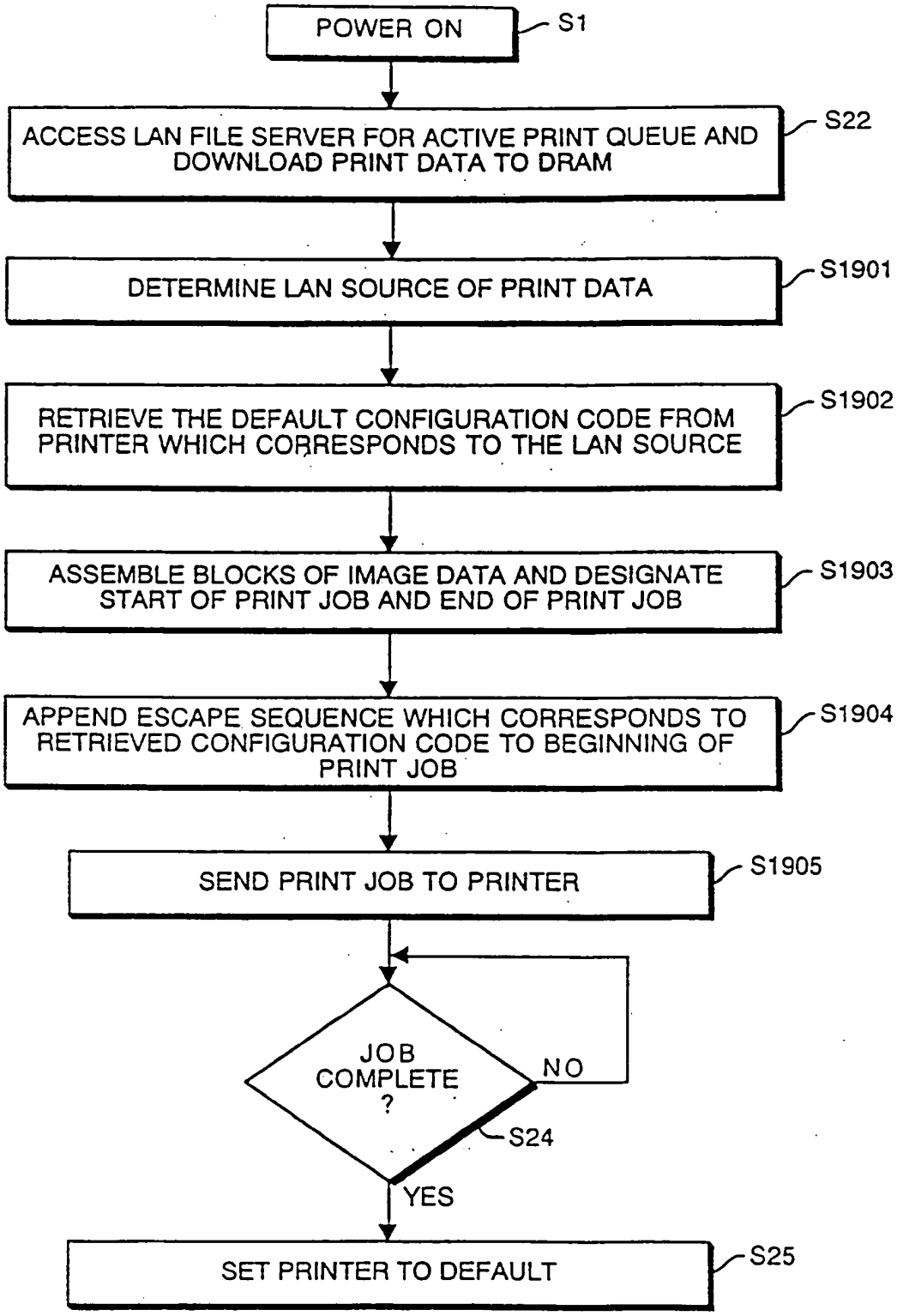

POWER ON
S1
ACCESS LAN FILE SERVER FOR ACTIVE PRINT QUEUE AND DOWNLOAD PRINT DATA TO DRAM
S22
DETERMINE LAN SOURCE OF PRINT DATA
S1901
RETRIEVE THE DEFAULT CONFIGURATION CODE FROM PRINTER WHICH CORRESPONDS TO THE LAN SOURCE
S1902
ASSEMBLE BLOCKS OF IMAGE DATA AND DESIGNATE START OF PRINT JOB AND END OF PRINT JOB
S1903
APPEND ESCAPE SEQUENCE WHICH CORRESPONDS TO RETRIEVED CONFIGURATION CODE TO BEGINNING OF PRINT JOB
S1904
SEND PRINT JOB TO PRINTER
S1905
JOB COMPLETE ?
NO
S24
YES
SET PRINTER TO DEFAULT
S25

FIG.19

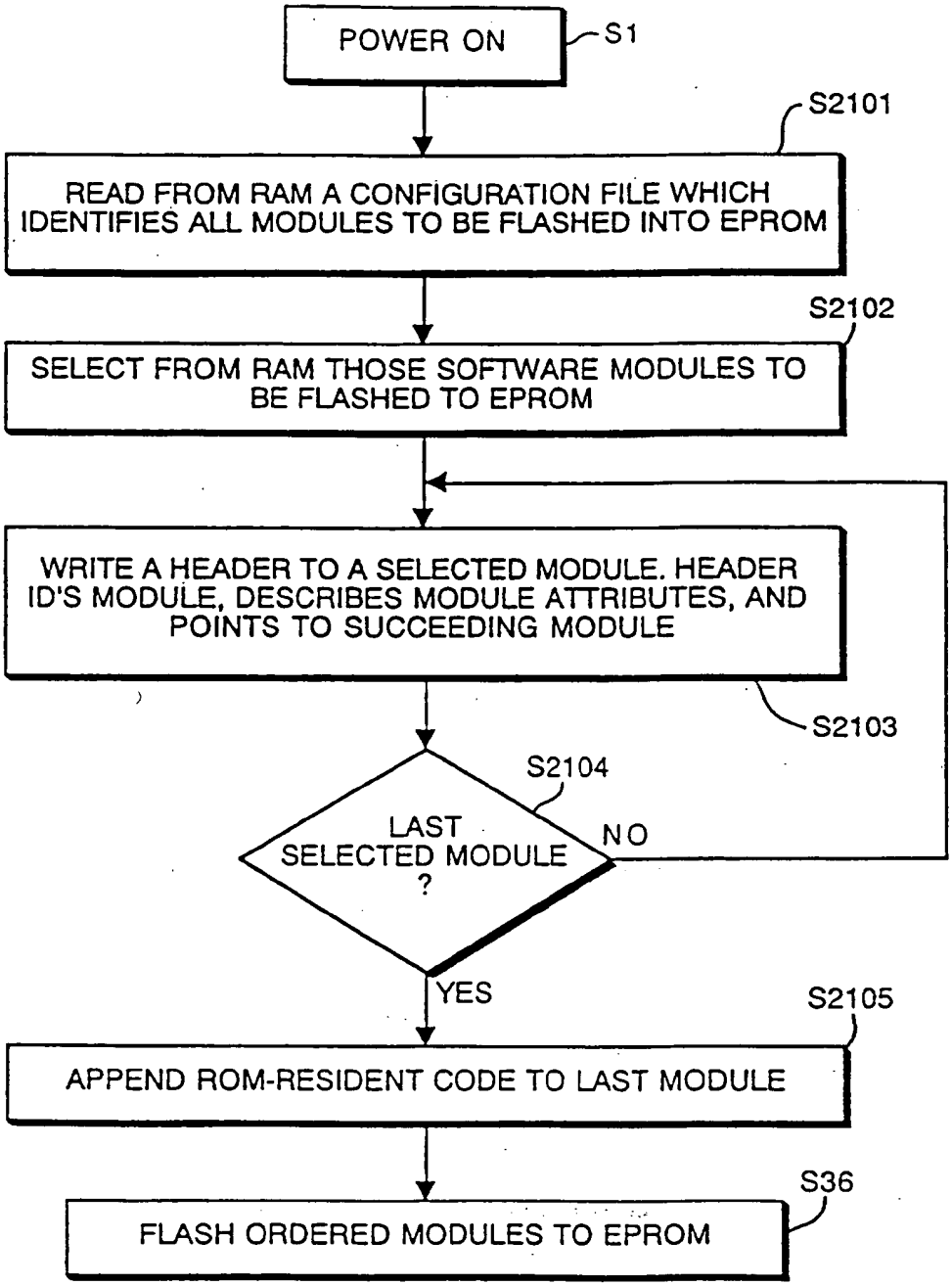

POWER ON
S1
S2101
READ FROM RAM A CONFIGURATION FILE WHICH IDENTIFIES ALL MODULES TO BE FLASHED INTO EPROM
S2102
SELECT FROM RAM THOSE SOFTWARE MODULES TO BE FLASHED TO EPROM
WRITE A HEADER TO A SELECTED MODULE. HEADER ID'S MODULE, DESCRIBES MODULE ATTRIBUTES, AND POINTS TO SUCCEEDING MODULE
S2103
S2104
LAST SELECTED MODULE ?
NO
YES
S2105
APPEND ROM-RESIDENT CODE TO LAST MODULE
S36
FLASH ORDERED MODULES TO EPROM

FIG.21

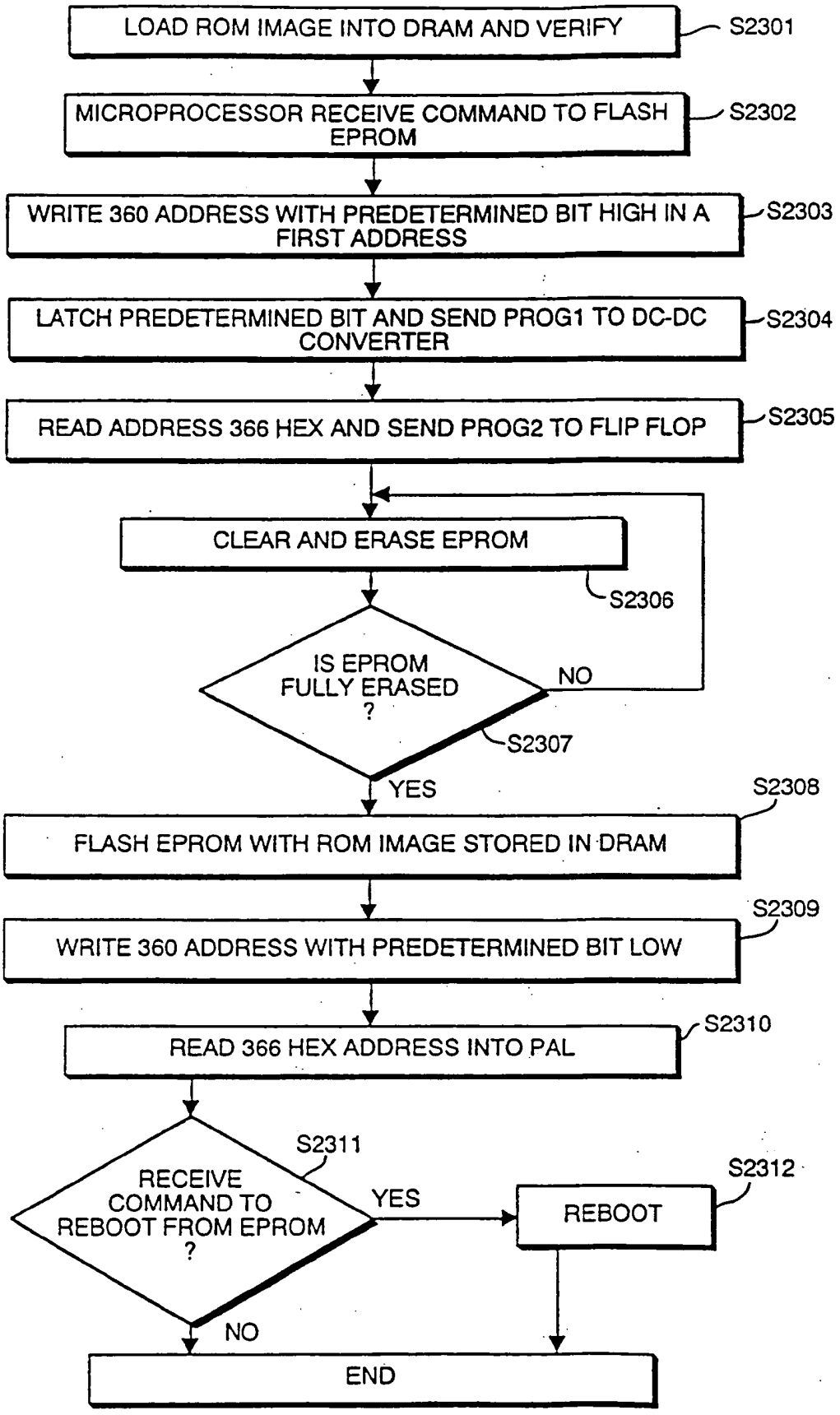

LOAD ROM IMAGE INTO DRAM AND VERIFY
S2301
MICROPROCESSOR RECEIVE COMMAND TO FLASH EPROM
S2302
WRITE 360 ADDRESS WITH PREDETERMINED BIT HIGH IN A FIRST ADDRESS
S2303
LATCH PREDETERMINED BIT AND SEND PROG1 TO DC-DC CONVERTER
S2304
READ ADDRESS 366 HEX AND SEND PROG2 TO FLIP FLOP
S2305
CLEAR AND ERASE EPROM
S2306
IS EPROM FULLY ERASED ?
NO
S2307
YES
S2308
FLASH EPROM WITH ROM IMAGE STORED IN DRAM
S2309
WRITE 360 ADDRESS WITH PREDETERMINED BIT LOW
S2310
READ 366 HEX ADDRESS INTO PAL
S2311
RECEIVE COMMAND TO REBOOT FROM EPROM ?
YES
S2312
REBOOT
NO
END

FIG.23

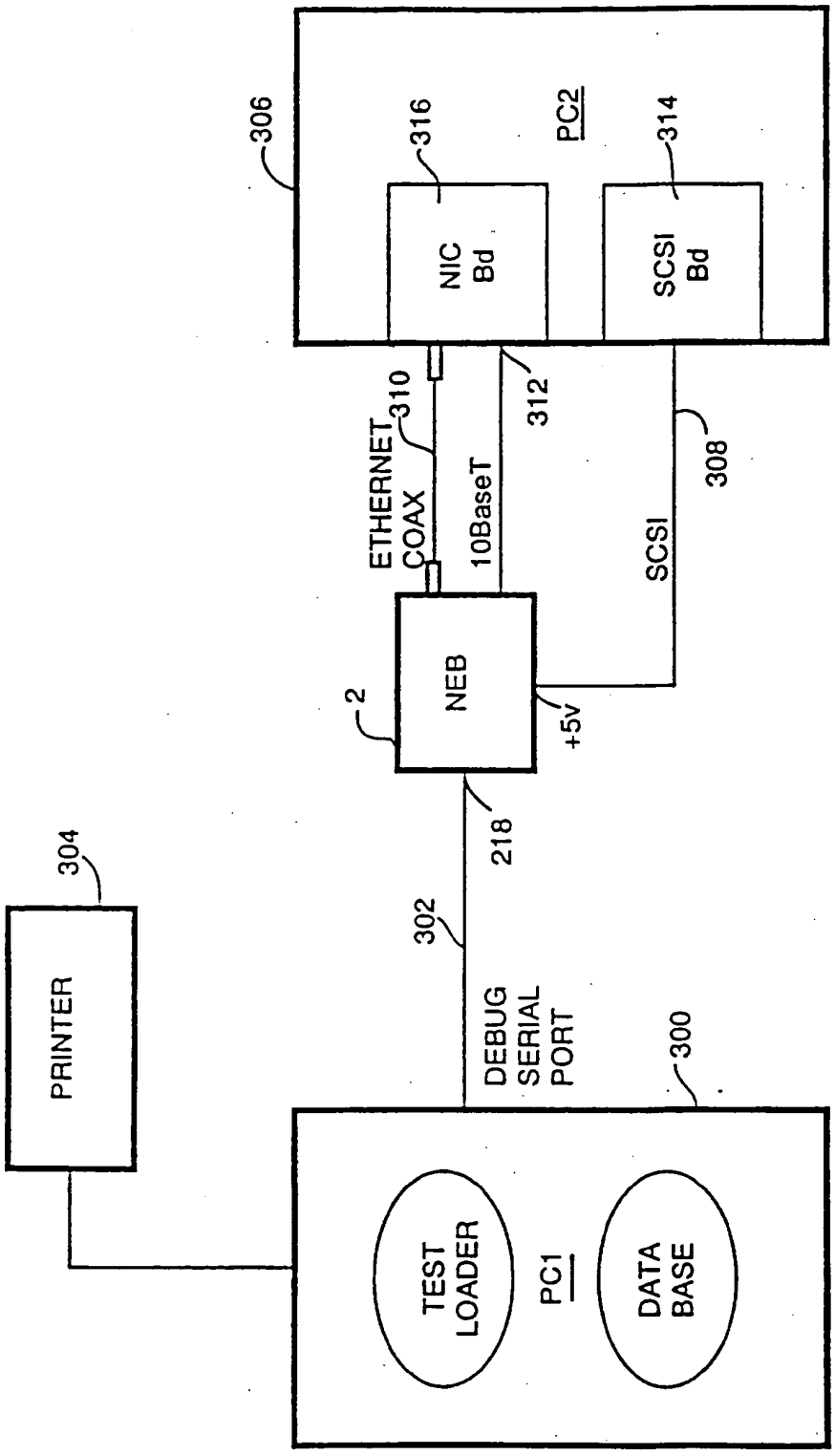

306
316
PC2
314
NIC Bd
SCSI Bd
310
312
308
ETHERNET COAX
10BaseT
SCSI
2
NEB
+5V
218
304
302
PRINTER
DEBUG SERIAL PORT
300
TEST LOADER
PC1
DATA BASE

FIG. 25

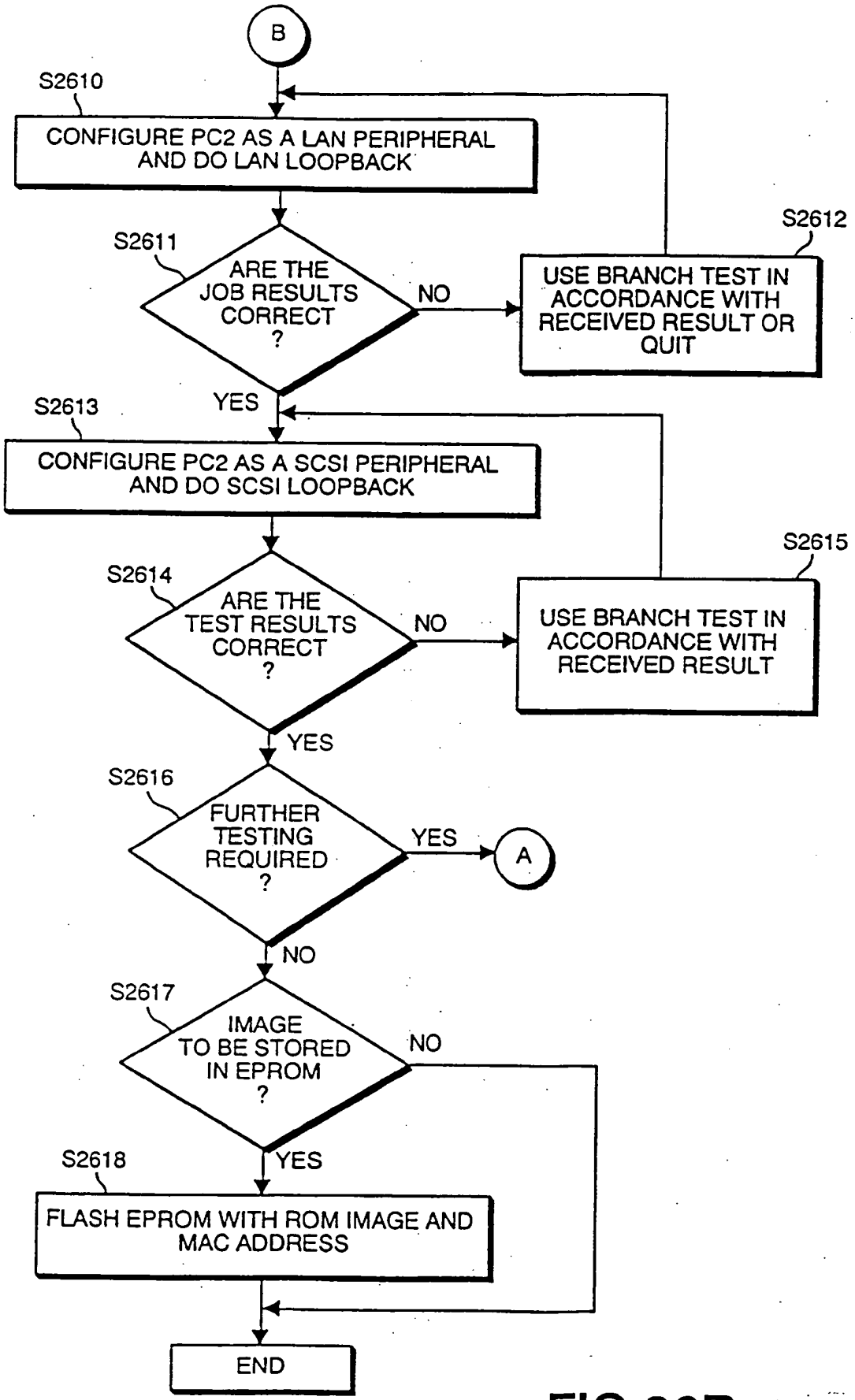

B
S2610
CONFIGURE PC2 AS A LAN PERIPHERAL AND DO LAN LOOPBACK
S2611
ARE THE JOB RESULTS CORRECT ?
NO
S2612
USE BRANCH TEST IN ACCORDANCE WITH RECEIVED RESULT OR QUIT
YES
S2613
CONFIGURE PC2 AS A SCSI PERIPHERAL AND DO SCSI LOOPBACK
S2614
ARE THE TEST RESULTS CORRECT ?
NO
S2615
USE BRANCH TEST IN ACCORDANCE WITH RECEIVED RESULT
YES
S2616
FURTHER TESTING REQUIRED ?
YES
A
NO
S2617
IMAGE TO BE STORED IN EPROM ?
NO
YES
S2618
FLASH EPROM WITH ROM IMAGE AND MAC ADDRESS
END

FIG.26B